

## APPENDIX A

FIGS. A1, A2 and A3 show possible implementations for the low-pass filter ( $L_p$ ) and the high-pass filter ( $H_p$ ) cut-off filters. The order of these particular filters is kept small to support a low-MIPS implementation. FIG. A4 shows a possible implementation for the high and low shelving equalizers. These equalizers are used for  $L_s$  and  $H_s$ . FIG. A5 shows a parametric equalizer implementation usable for boost or cut applications. Combinations of such equalizers can be combined, as shown in FIGS. A6 and A7, to build the compensated notch effect.

0001

## APPENDIX B

Figure B1 is a block diagram of a compensation system used to compensate a small bookshelf speaker having a 5-inch bass driver and 3-inch tweeter. The speaker is characterized as follows:

Woofer:      dome = 4.33 k (notch with LC)  
                 cone = 4.9 k (compensated notch only)  
                 edge of cone = 2.8 k (active notch)

Tweeter:      dome = 15.1 k (no compensation)  
                 cone cr = 11.6 k (active notch)  
                 cone cr = 9.32 k / 9.45 k (damp compound)  
                  $W_0 = 1.38$  k (LCR notch)

Fourteen adjustments are provided. FIG. B2 shows circuits used to create  $L_x$ ,  $H_x$ ,  $L_s$ ,  $H_s$ ,  $L_p$ , and  $H_p$  filters and parameters adjustments. A lower frequency cutoff peak was added to create response character of an 8 inch bookshelf system. For this circuit, Q and peaking amplitude are set by the components marked by square boxes. FIGS. B3 and B4 show  $W_0$  notch sections and representative tuning for the speaker.  $W_0 \dots$  and  $|A|$  in dB are adjustable, while a resistor set Q. Two groups of these make four adjustments. FIG. B5 shows a single low-Q boost. Adjustments for boost and frequency are provided by the op amp section. FIG. B6 is an all-pass equalizer used for time correction. The all-pass equalizer combines outputs from the six active process circuits.

## APPENDIX C

A computer adjusted DSP compensator was implemented using a standard PC and a Motorola EVM56362 DSP evaluation board. Source code and application notes follow.

## Speaker Tuning Application Note

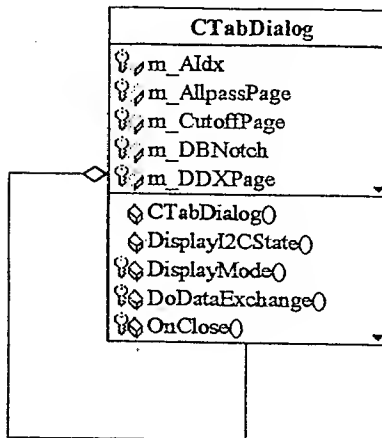
The speaker tuning application is written as a Windows OS 32-bit application using the object-oriented MFC application framework (please refer to the documentation of MFC that is included with Microsoft Visual C++ development environment). The application has a dialog interface. The main dialog class, CTabDialog, is a subclass of the MFC CDialog class. The CTabDialog class implements a "tabbed" dialog interface. Each tab in the dialog is a subclass of the MFC CPropertyPage class. Each tab represents different aspects of the speaker correction algorithm. The following is a list of tab classes:

- CMainPage which implements UI for pre and post volume controls amongst other things
- CShelvPage which implements UI controls for low and high shelving equalization filters
- CCutoffPage which implements UI controls for low and high peaking cutoff filters
- CNotchPage1 and CNotchPage2 which implements UI controls for a number of notch filters (to for example limit resonance in the speaker)
- CStWaveRejectPage which implements UI controls for a set of filters which can limit standing waves in the speaker cabinet
- CDBNotch which implements UI controls for a double-tuned notch filter
- CAllpassPage which implements UI for a 2<sup>nd</sup>-order allpass filter

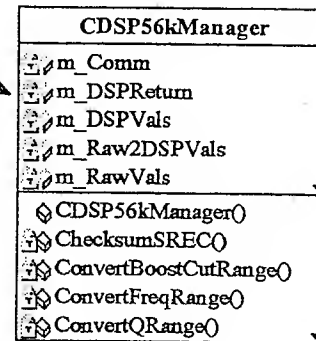
Each adjustable parameter on a tab page is represented by an instance of the CSlider class. Each UI slider has a range of 4096 discrete values. When a user manipulates a UI slider a message is sent from the tab page to an instance of the CDSP56Manager class. In this class, the appropriate calculation takes place to transform the linear input value into one or more values necessary to compute the transfer function represented by the user settings. These computed values are transmitted to the DSP using an I2C serial connection. The DSP executes the calculations necessary for the real-time implementation of the above-mentioned transfer function. The DSP is capable of computing a series of filter calculations in real-time to allow the total cascaded transfer function of all the speaker correction filters to be realized. The DSP can receive analog or digital input data and transmit processed analog or digital output data.

# Speaker Parameter File Input and Output Logic

The tab dialog object instance receives messages (from the Windows OS) in response to user action in the application menus

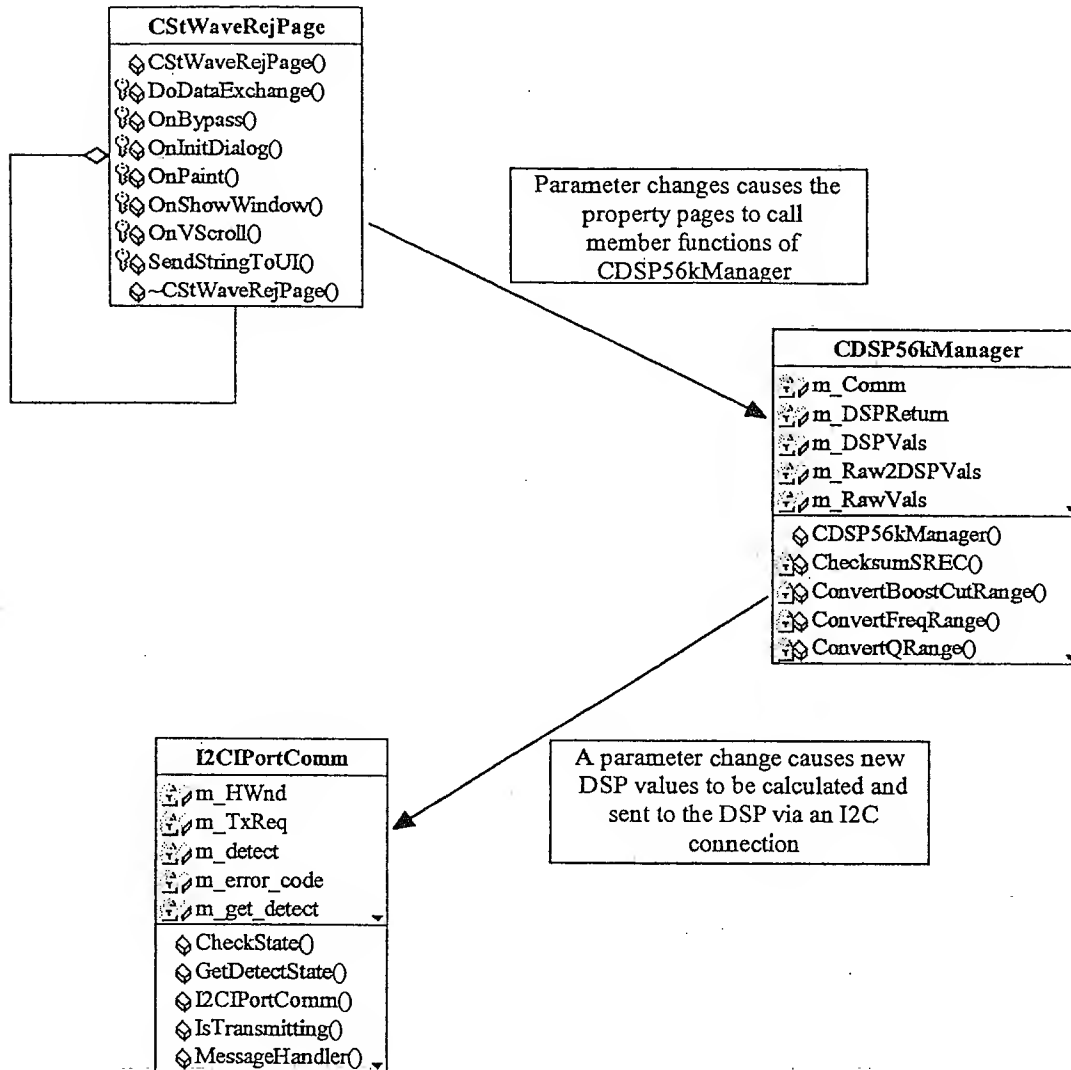


The user "open" or "save" request results in function calls to member functions of the CDSP56kManager. These calls can restore or return the state of all DSP parameters.



# Example of Property Page UI to DSP connection logic

The property pages of the TabDialog receive mapped messages (from the Windows OS) in response to user actions



```

// COMPortChooser.cpp : Implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "COMPortChooser.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// CCOMPortChooser dialog

CCOMPortChooser::CCOMPortChooser(CWnd* pParent /*=NULL*/)
: CDialog(CCOMPortChooser::IDD, pParent)
{
   //{{AFX_DATA_INIT(CCOMPortChooser)
    // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
}

void CCOMPortChooser::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CCOMPortChooser)
    // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CCOMPortChooser, CDialog)
   //{{AFX_MSG_MAP(CCOMPortChooser)
    ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
    ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
    ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
    ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

//////////////////////////////////////
// CCOMPortChooser message handlers

void CCOMPortChooser::OnRadio1()
{
    // TODO: Add your control notification handler code here
    which_port = 1;
}

void CCOMPortChooser::OnRadio2()
{
    // TODO: Add your control notification handler code here
    which_port = 2;
}

void CCOMPortChooser::OnRadio3()
{
    // TODO: Add your control notification handler code here
    which_port = 3;
}

void CCOMPortChooser::OnRadio4()
{
    // TODO: Add your control notification handler code here
    which_port = 4;
}

```

```

BOOL CCOMPortChooser::OnInitDialog()
{

```

```
CDialog::OnInitDialog();
```

```
which_port = 3;
```

```
return TRUE; // return TRUE unless you set the focus to a control
// EXCEPTION: ctrl+enter sets focus
```

```
// EXCEPTION: OCX Property Pages should return FALSE
```



```
// CutoffPage.cpp : implementation file
//
```

```
#include "stdafx.h"
#include "sa.h"
#include "CutoffPage.h"
#include "DSP56kManager.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
////////////////////////////////////
// CCutoffPage property page
```

```
IMPLEMENT_DYNCREATE(CCutoffPage, CPropertyPage)
```

```
CCutoffPage::CCutoffPage() : CPropertyPage(CCutoffPage::IDD)
{
    ///{{AFX_DATA_INIT(CCutoffPage)
    ///}}AFX_DATA_INIT
}
```

```
CCutoffPage::~CCutoffPage()
{
}
```

```
void CCutoffPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    ///{{AFX_DATA_MAP(CCutoffPage)
    DDX_Control(pDX, IDC_CHECK6, m_BypassSecondButton);
    DDX_Control(pDX, IDC_CHECK5, m_BypassFirstButton);
    DDX_Control(pDX, IDC_SLIDER6, m_HiBoostSlider);
    DDX_Control(pDX, IDC_SLIDER5, m_HiQSlider);
    DDX_Control(pDX, IDC_SLIDER4, m_HiFreqSlider);
    DDX_Control(pDX, IDC_SLIDER3, m_LoBoostSlider);
    DDX_Control(pDX, IDC_SLIDER2, m_LoQSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_LoFreqSlider);
    ///}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CCutoffPage, CPropertyPage)
    ///{{AFX_MSG_MAP(CCutoffPage)
    ON_WM_VSCROLL()
    ON_BN_CLICKED(IDC_CHECK5, OnBypassFirst)
    ON_BN_CLICKED(IDC_CHECK6, OnBypassSecond)
    ///}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////
// CCutoffPage message handlers
```

```
BOOL CCutoffPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_LoFreqSlider.SetRange(0, CONTROL_RANGE);
    m_LoFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Freq));
    m_LoFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_LoQSlider.SetRange(0, CONTROL_RANGE);
    m_LoQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Q));
    m_LoQSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_LoBoostSlider.SetRange(0, CONTROL_RANGE);
    m_LoBoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Boost));
    m_LoBoostSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_HiFreqSlider.SetRange(0, CONTROL_RANGE);
    m_HiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Freq));
```

0007

```

m_HiFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_HiQSlider.SetRange(0,CONTROL_RANGE);
m_HiQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Q));
m_HiQSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_HiBoostSlider.SetRange(0,CONTROL_RANGE);
m_HiBoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Boost));
m_HiBoostSlider.SetTicFreq((CONTROL_RANGE+1)/16);

// g_DSPManager->SetCutoff1Freq(CONTROL_RANGE-m_LoFreqSlider.GetPos());
// g_DSPManager->SetCutoff1Q(CONTROL_RANGE-m_LoQSlider.GetPos());
// g_DSPManager->SetCutoff1Boost(CONTROL_RANGE-m_LoBoostSlider.GetPos());
// g_DSPManager->SetCutoff2Freq(CONTROL_RANGE-m_HiFreqSlider.GetPos());
// g_DSPManager->SetCutoff2Q(CONTROL_RANGE-m_HiQSlider.GetPos());
// g_DSPManager->SetCutoff2Boost(CONTROL_RANGE-m_HiBoostSlider.GetPos());

return TRUE; // return TRUE unless you set the focus to a control
            // EXCEPTION: OCX Property Pages should return FALSE
}

void CCutoffPage::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if( (CSliderCtrl *)pScrollBar == &m_LoFreqSlider )
        g_DSPManager->SetCutoff1Freq(CONTROL_RANGE-m_LoFreqSlider.GetPos());
    else if( (CSliderCtrl *)pScrollBar == &m_LoQSlider )
        g_DSPManager->SetCutoff1Q(CONTROL_RANGE-m_LoQSlider.GetPos());
    else if( (CSliderCtrl *)pScrollBar == &m_LoBoostSlider )
        g_DSPManager->SetCutoff1Boost(CONTROL_RANGE-m_LoBoostSlider.GetPos());
    else if( (CSliderCtrl *)pScrollBar == &m_HiFreqSlider )
        g_DSPManager->SetCutoff2Freq(CONTROL_RANGE-m_HiFreqSlider.GetPos());
    else if( (CSliderCtrl *)pScrollBar == &m_HiQSlider )
        g_DSPManager->SetCutoff2Q(CONTROL_RANGE-m_HiQSlider.GetPos());
    else if( (CSliderCtrl *)pScrollBar == &m_HiBoostSlider )
        g_DSPManager->SetCutoff2Boost(CONTROL_RANGE-m_HiBoostSlider.GetPos());
}

void CCutoffPage::OnBypassFirst()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassFirstButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassHipass);
}

void CCutoffPage::OnBypassSecond()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassSecondButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassLopass);
}

```

```

#if !defined(AFX_COMPORTCHOOSER_H__B5D510E7_F7D5_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_COMPORTCHOOSER_H__B5D510E7_F7D5_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// COMPortChooser.h : header file
//

////////////////////////////////////

// CCOMPortChooser dialog

class CCOMPortChooser : public CDialog
{
// Construction
public:
    CCOMPortChooser(CWnd* pParent = NULL);    // standard constructor

    int which_port;

// Dialog Data
    //{AFX_DATA(CCOMPortChooser)
    enum { IDD = IDD_DIALOG3 };
        // NOTE: the ClassWizard will add data members here
    //}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CCOMPortChooser)
    protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
    protected:

        // Generated message map functions
        //{AFX_MSG(CCOMPortChooser)
        afx_msg void OnRadio1();
        afx_msg void OnRadio2();
        afx_msg void OnRadio3();
        afx_msg void OnRadio4();
        virtual BOOL OnInitDialog();
        //}AFX_MSG
        DECLARE_MESSAGE_MAP()
    };

    //{AFX_INSERT_LOCATION}}
    // Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_COMPORTCHOOSER_H__B5D510E7_F7D5_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0009

```

#if !defined(AFX_CUTOFFPAGE_H__6F151625_D84D_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_CUTOFFPAGE_H__6F151625_D84D_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// CutoffPage.h : header file
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CCutoffPage dialog

class CCutoffPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CCutoffPage)

// Construction
public:
    CCutoffPage();
    ~CCutoffPage();

// Dialog Data
    //{AFX_DATA(CCutoffPage)
    enum { IDD = IDD_PP7 };
    CButton m_BypassSecondButton;
    CButton m_BypassFirstButton;
    CSliderCtrl m_HiBoostSlider;
    CSliderCtrl m_HiQSlider;
    CSliderCtrl m_HiFreqSlider;
    CSliderCtrl m_LoBoostSlider;
    CSliderCtrl m_LoQSlider;
    CSliderCtrl m_LoFreqSlider;
    //}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{AFX_VIRTUAL(CCutoffPage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{AFX_MSG(CCutoffPage)
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnBypassFirst();
    afx_msg void OnBypassSecond();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_CUTOFFPAGE_H__6F151625_D84D_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0010

```

#if !defined(AFX_ALLPASSPAGE_H_F4767B24_11B3_11D3_96EE_006097CDB9E2__INCLUDED_)
#define AFX_ALLPASSPAGE_H_F4767B24_11B3_11D3_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// AllpassPage.h : header file
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CallpassPage dialog

class CTabDialog;

class CallpassPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CallpassPage)

// Construction
public:
    CallpassPage();
    ~CallpassPage();

// Dialog Data
    //{{AFX_DATA(CallpassPage)
    enum { IDD = IDD_PP10 };
    CButton m_BypassButton;
    CSliderCtrl m_QSlider;
    CSliderCtrl m_FrequencySlider;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CallpassPage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CallpassPage)
    afx_msg void OnBypass();
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnPaint();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_ALLPASSPAGE_H_F4767B24_11B3_11D3_96EE_006097CDB9E2__INCLUDED_)

```

```

// AllpassPage.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "AllpassPage.h"
#include "TabDialog.h"
#include "DSP56kManager.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CallpassPage property page

IMPLEMENT_DYNCREATE(CallpassPage, CPropertyPage)

CallpassPage::CallpassPage() : CPropertyPage(CallpassPage::IDD)
{
    //{AFX_DATA_INIT(CallpassPage)
    //}AFX_DATA_INIT
}

CallpassPage::~CallpassPage()
{
}

void CallpassPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{AFX_DATA_MAP(CallpassPage)
    DDX_Control(pDX, IDC_CHECK1, m_BypassButton);
    DDX_Control(pDX, IDC_SLIDER3, m_QSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_FrequencySlider);
    //}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CallpassPage, CPropertyPage)
    //{AFX_MSG_MAP(CallpassPage)
    ON_BN_CLICKED(IDC_CHECK1, OnBypass)
    ON_WM_VSCROLL()
    ON_WM_PAINT()
    //}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CallpassPage message handlers

void CallpassPage::OnBypass()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassAllpass);
}

BOOL CallpassPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_FrequencySlider.SetRange(0, CONTROL_RANGE);
    m_FrequencySlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAllpassFreq));
    m_FrequencySlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_QSlider.SetRange(0, CONTROL_RANGE);
    m_QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAllpassQ));
    m_QSlider.SetTicFreq((CONTROL_RANGE+1)/16);
}

```

```

m_ParentWindow = (CTabDialog *) GetParent()->GetParent();

return TRUE; // return TRUE unless you set the focus to a control
// EXCEPTION: OCX Property Pages should return FALSE
}

void CallpassPage::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if( slider == &m_FrequencySlider )
        which = kAllpassFreq;
    else if( slider == &m_QSlider )
        which = kAllpassQ;
    else
        return;

    g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(), which);
    SendStringToUI(which);
}

void CallpassPage::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which, str);
    m_ParentWindow->SetStatusString(0, str);
}

void CallpassPage::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_FrequencySlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAllpassFreq));
    m_QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAllpassQ));
    m_BypassButton.SetCheck(g_DSPManager->GetBypassSection(kBypassAllpass));

    // Do not call CPropertyPage::OnPaint() for painting messages
}

```

```

// DSPComm.h: interface the DSPComm class.
//
////////////////////////////////////

#ifndef AFX_DSPCOMM_H_33F9EF05_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_
#define AFX_DSPCOMM_H_33F9EF05_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class DSPComm
{
public:
    virtual BOOL IsTransmitting();
    virtual long GetDetectState(void);
    virtual BOOL CheckState(void);
    virtual long SendDSPMemory(char *, long);
    virtual long SendDSPWord(long);
    DSPComm();
    virtual ~DSPComm();
};

#endif // !defined(AFX_DSPCOMM_H_33F9EF05_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_)

```



```

// DSPComm.cpp: implementation of the DSPComm class.
//
/////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sa.h"
#include "DSPComm.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

/////////////////////////////////////////////////////////////////
// Construction/Destruction
/////////////////////////////////////////////////////////////////

DSPComm::DSPComm()
{
}

DSPComm::~DSPComm()
{
}

long DSPComm::SendDSPWord(long)
{
    return( 0L );
}

long DSPComm::SendDSPMemory(char *data, long len)
{
    return( 0L );
}

BOOL DSPComm::CheckState()
{
    return( true );
}

long DSPComm::GetDetectState()
{
    return( 0L );
}

BOOL DSPComm::IsTransmitting()
{
    return( false );
}

```

0015

```

// DSP56kManager.h: interface for the CDSP56kManager class.
//
////////////////////////////////////

#ifndef AFX_DSP56KMANAGER_H__0CF7E204_D790_11D2_96EE_006097CDB9E2__INCLUDED_
#define AFX_DSP56KMANAGER_H__0CF7E204_D790_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "DSP56Equates.h"
#include "DSPComm.h"

#define CONTROL_RANGE_L2 12
#define CONTROL_RANGE ((1L<<CONTROL_RANGE_L2)-1)
#define PG_CONTROL_AMT (CONTROL_RANGE/64)
#define CONTROL_RANGE_SC (23-CONTROL_RANGE_L2)
#define DSP_CONTROL_MAX 8388608.

class CDSP56kManager
{
    long m_DSPReturn;
    long m_Raw2DSPVals[kUIArraySize];
    long m_RawVals[kUIArraySize];
    long m_DSPVals[kDSPArraySize];
    double m_pi;
    DSPComm *m_Comm;

public:
    void GetPureStringValue(int which, CString &str);
    void GetFilterBlob(CStringArray &array);
    virtual BOOL GetBypassSection(int which);
    virtual void SetBypassSection(BOOL value, int which);
    virtual BOOL IsBusy();
    virtual void GetStringValue(int which, CString &str);
    virtual BOOL IsReady(void);
    virtual void SetAnalogInput(long);
    virtual int GetHDCDMode(void);
    virtual void ResetAll(void);
    virtual void SetHDCDGainScale(long);
    virtual void SetHDCDBypass(long);
    virtual void SetDDXCompBypass(long);
    virtual void SetBypass(long value);
    virtual void GetDSPSettings(CStringArray &array);
    virtual void SetDSPSettings(CStringArray &array);
    virtual void SetParamValue(long value, long which);
    virtual long GetParamValue(long which);
    void DownloadDSPCode(void);

    CDSP56kManager(HWND p);
    virtual ~CDSP56kManager();

private:
    void SetNotchQ(long value, long which, long freqwhich);
    void SetNotchFreq(long value, long which);
    void SetShelvFreq(long value, long which);
    void SetBoostCut(long value, long which);
    void SetRawValue(long value, long which);
    double DoConvertFreqRange(double in, double top, double bottom);
    double ConvertFreqRange(int which);
    double ConvertQRange(long val);
    double ConvertBoostCutRange(long val);
    virtual void SendDSPValue(long which);

    int ChecksumSREC(char *lineptr, int N);
    void GetSRecordAddressRange(char *s, long *start, long *end, char **data);

protected:
    virtual void SetDelay(long value);
    void SetLoCutoff(void);

```

```

void SetHiCutoff(void);
void SetHiCutoff2(void);

virtual void SetHiCutoff2Q(long value);
virtual void SetHiCutoff2Freq(long value);
virtual void SetHiCutoffQ(long value);
virtual void SetHiCutoffFreq(long value);
virtual void SetLoCutoffQ(long value);
virtual void SetLoCutoffFreq(long value);
virtual void SetPreVolume(long value);
virtual void SetAnalogVolume(long value);

// virtual void SetCutoff1Freq(long value);
// virtual void SetCutoff1Q(long value);
// virtual void SetCutoff1Boost(long value);
// virtual void SetCutoff2Freq(long value);
// virtual void SetCutoff2Q(long value);
// virtual void SetCutoff2Boost(long value);
virtual void SetShelv1Freq(long value);
virtual void SetShelv1Boost(long value);
virtual void SetShelv2Freq(long value);
virtual void SetShelv2Boost(long value);
virtual void SetNotch1Cut(long value);
virtual void SetNotch1Q(long value);
virtual void SetNotch1Freq(long value);
virtual void SetNotch2Cut(long value);
virtual void SetNotch2Q(long value);
virtual void SetNotch2Freq(long value);
virtual void SetNotch3Cut(long value);
virtual void SetNotch3Q(long value);
virtual void SetNotch3Freq(long value);
virtual void SetNotch4Cut(long value);
virtual void SetNotch4Q(long value);
virtual void SetNotch4Freq(long value);
virtual void SetNotch5Q(long value);
virtual void SetNotch5Boost(long value);
virtual void SetNotch6Cut(long value);
virtual void SetNotch6Q(long value);
virtual void SetNotch6Freq(long value);
virtual void SetNotch7Cut(long value);
virtual void SetNotch7Q(long value);
virtual void SetNotch7Freq(long value);
virtual void SetNotch9Freq(long value);

virtual void SetMainVolume(long value);

};

extern CDSP56kManager *g_DSPManager;

#endif // !defined(AFX_DSP56KMANAGER_H__0CF7E204_D790_11D2_96EE_006097CDB9E2__INCLUDED_)

```

```

// DSP56kManager.cpp: implementation of the CDSP56kManager class.
//
/////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sa.h"
#include "DSP56kManager.h"
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#if 0
#include "unit_ppi.h"
#include "functs.h"
#endif

#ifdef SPI
#include "SPIPEMicroComm.h"
#endif
#include "I2CIPortComm.h"

#include "sapmem.h"
#include "SSTParams.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

#if 0
static int cmd_delay = 1000;
static int io_delay = 50;
#endif

#define FILE_VERSION 2 // file version
#define GAIN_SCALE 8.

// volume
#define VOLTOP 1.
#define VOLBOT 3.909e-6
#define VOLRANGE (VOLTOP-VOLBOT)

// hi and lo cutoff and shelving ranges
#define FLCTOPRANGE 0.149 // 1000/22050
#define FLCBOTRANGE 0.001769 // 10/22050
#define FLCRANGE (FLCTOPRANGE-FLCBOTRANGE)

#define FLCTOPRANGE2 4.255 // 2000
#define FLCBOTRANGE2 2.134 // 15

#define FHCTOPRANGE 1 // 22050/22050
#define FHCBOTRANGE 0.63 // 8000/22050
#define FHCBOTRANGE2 0.42 // 4000/22050
#define FHCRANGE (FHCTOPRANGE-FHCBOTRANGE)
#define FHCRANGE2 (FHCTOPRANGE-FHCBOTRANGE2)

#define QLCTOPRANGE 1.0 // 1
#define QLCBOTRANGE 0.003891 // 0.001
#define QLCRANGE (QLCTOPRANGE-QLCBOTRANGE)

#define SAMPLING_FREQ 44100
#define DELAY_LENGTH 128.

// converts frequency range to log scale
#define FTOPRANGE 1.
#define FBOTRANGE 0.134
#define FRANGE (FTOPRANGE-FBOTRANGE)
#define FTOPRANGE2 0.585 // about 2205 top.
#define FBOTRANGE2 3.531E-3

```

```

// new ranges using actual frequencies
#define FTOPRANGE3 4.431 // 3000 Hz
#define FBOTRANGE3 2.433 // 30 Hz

// hi cutoff filter ranges
#define QHC2BOT 0.74
#define QHC2TOP 2.258
#define FHC2BOT 3.954 // 1000 Hz
#define FHC2TOP 5.255 // 20 kHz

#define FHCTOPRANGE3 5.13 // 15000
#define FHCBOTRANGE3 FHC2BOT

```

```

static char *m_ParamNames[kUIArraySize] = { "MainVolume",
    "LoShelfFreq",
    "LoShelfGain",
    "HiShelfFreq",
    "HiShelfGain",
    "Notch1Freq",
    "Notch1Q",
    "Notch1Cut",
    "Notch2Freq",
    "Notch2Q",
    "Notch2Cut",
    "Notch3Freq",
    "Notch3Q",
    "Notch3Cut",
    "Notch4Freq",
    "Notch4Q",
    "Notch4Cut",
    "Notch5Freq",
    "Notch5Q",
    "Notch5Cut",
    "PreVolume",
    "Bypass",
    "LoCutoffFreq",
    "LoCutoffQ",
    "HiCutoffFreq",
    "HiCutoffQ",
    "HDCDBypass",
    "HDCDGainScaleOff",
    "DDXCompBypass",
    "Notch6Freq",
    "Notch6Q",
    "Notch6Cut",
    "Notch7Freq",
    "Notch7Q",
    "Notch7Cut",
    "AnalogIn",
    "AnalogVolume",
    "Delay",
    "AllpassFreq",
    "AllpassQ",
    "BypassMask",
    "HiCutoff2Freq",
    "HiCutoff2Q",
    "Notch8Freq",
    "Notch8Q",
    "Notch8Cut",
    "Notch9Freq",
    "Notch9Q",
    "Notch9Cut",
    "NotchAFreq",
    "NotchAQ",
    "NotchACut",
    "DBNotchWidth" };

```

```

////////////////////////////////////
// SREC code
////////////////////////////////////

```

0018

```
int CDSP56kManager::ChecksumSREC(char *lineptr, int N)
{
```

```
    unsigned char curbyte, sum=0xff;
    for (int x=2; x<N-2; x+=2)
    {
        sscanf(lineptr+x, "%2x", &curbyte);
        sum-=curbyte;
    }
```

```
    // last one is checksum
    sscanf(lineptr+x, "%2x", &curbyte);
    return (curbyte==sum);
}
```

```
void CDSP56kManager::GetSRecordAddressRange(char *s, long *start, long *end, char **data)
{
```

```
    long address, count, soffset;
```

```
    *start = -1L;
    *end = -1L;
```

```
    if (*s != 'S') // not a srec
        return;
```

```
    // Let's check the checksum for this line
    ASSERT ( ChecksumSREC(s, strlen(s)) );
    switch( s[1] )
```

```
    {
        case '1':
            soffset=8;
            sscanf(&(s[4]), "%4x", &address);
            break;
        case '2':
            soffset=10;
            sscanf(&(s[4]), "%6x", &address);
            break;
        case '3':
            soffset=12;
            sscanf(&(s[4]), "%8x", &address);
            break;
        default:
            return;
            break;
    }
```

```
    count = strlen(s) - soffset - 2;
    *start = address;
    *end = address + (count/6);    // each word is 6 characters
    *data = s + soffset;
```

```
}
```

```
void CDSP56kManager::DownloadDSPCode()
{
```

```
    char **r = pmem;
    int count = PRECORDS;
    long start, end;
    long ts, te;
    char **recs;
    char *srecdata, *td;
    char *data, *dptr, *rdata, c;
```

```
    long size, bsize;
    long taddress, recstart, recend;
    long value;
```

0019

```

// get starting and ending addresses
start = 0x7FFFFFFFL;
end = 0;
recs = r;
for( int i = 0; i < count ; i++ )
{
    GetSRecordAddressRange(*recs++, &ts, &te, &srecdata);
    if( ts < 0L ) continue;
    start = min(start, ts);
    end = max(end, te);
}

size = end - start;
bsize = size * 3;
rdata = (char *) malloc(bsize+6);    // 3 bytes per word plus length and address
ASSERT( data != NULL );
data = rdata + 6;

dptr = data;
for( i = 0 ; i < bsize ; i++ )    // clear memory
    *dptr++ = 0;
taddress = start;
while( taddress < end )
{
    recstart = 0x7FFFFFFFL;
    recs = r;
    for( int i = 0; i < count ; i++ )    // search for next address
    {
        GetSRecordAddressRange(*recs, &ts, &te, &td);
        if( ts >= 0L )
        {
            if( ts >= taddress )
            {
                if( ts < recstart )
                {
                    recstart = ts;
                    recend = te;
                    srecdata = td;
                }
            }
        }
        recs++;
    }
    ASSERT( recstart <= end );
    // copy record
    dptr = data + ((recstart - start)*3);
    while( recstart != recend )
    {
        sscanf(srecdata, "%2x", &value);
        *dptr++ = (char ) value;
        srecdata += 2;
        sscanf(srecdata, "%2x", &value);
        *dptr++ = (char ) value;
        srecdata += 2;
        sscanf(srecdata, "%2x", &value);
        *dptr++ = (char ) value;
        srecdata += 2;
        recstart++;
    }
    taddress = recend;
}

// copy size and address
dptr = rdata;
c = (size >> 16) & 0xFF;
*dptr++ = c;
c = (size >> 8) & 0xFF;
*dptr++ = c;
c = size & 0xFF;
*dptr++ = c;
c = (start >> 16) & 0xFF;

```

0020

```

        *dptr++ = c;
        c = (start >> 8) & 0xFF;
        *dptr++ = c;
        c = start & 0xFF;
        *dptr++ = c;

        // send data to DSP
        //m_Comm->SendDSPWord(size);
        //m_Comm->SendDSPWord(start);
        m_Comm->SendDSPMemory(rdata, bsize+6);

        free( rdata );
    }

    //////////////////////////////////////
    // Construction/Destruction
    //////////////////////////////////////

    CDSP56kManager::CDSP56kManager( HWND p )
    {

    #if 0
        if( load_dll()==false )
        {
            MessageBox(NULL, "UNIT_PPI not found in the system directory. Fix this and restart application.", "Danger,
            anger Will Robinson", MB_OK | MB_ICONSTOP );
        }
        else
        {
            init_port_spi(1);
            set_cmd_delay_cnt_value(cmd_delay);
            set_io_delay_cnt_value(io_delay);
        }
    #endif

        m_pi = acos(-1.);

        m_Comm = new I2CIPortComm(p);

        // DownloadDSPCode();

        int i;
        for( i = 0 ; i < kUIArraySize ; i++ )
        {
            m_Raw2DSPVals[i] = i;
        }

        m_Raw2DSPVals[kNotch6Freq] = kDSPNotch6Freq;
        m_Raw2DSPVals[kNotch6Q] = kDSPNotch6Q;
        m_Raw2DSPVals[kNotch6Cut] = kDSPNotch6Cut;
        m_Raw2DSPVals[kNotch7Freq] = kDSPNotch7Freq;
        m_Raw2DSPVals[kNotch7Q] = kDSPNotch7Q;
        m_Raw2DSPVals[kNotch7Cut] = kDSPNotch7Cut;
        m_Raw2DSPVals[kDelay] = kDSPDelay;
        m_Raw2DSPVals[kAllpassFreq] = kDSPAllpassFreq;
        m_Raw2DSPVals[kAllpassQ] = kDSPAllpassQ;

        m_Raw2DSPVals[kNotch8Freq] = kDSPNotch8Freq;
        m_Raw2DSPVals[kNotch8Q] = kDSPNotch8Q;
        m_Raw2DSPVals[kNotch8Cut] = kDSPNotch8Cut;
        m_Raw2DSPVals[kNotch9Freq] = kDSPNotch9Freq;
        m_Raw2DSPVals[kNotch9Q] = kDSPNotch9Q;
        m_Raw2DSPVals[kNotch9Cut] = kDSPNotch9Cut;
        m_Raw2DSPVals[kNotchAFreq] = kDSPNotchAFreq;
        m_Raw2DSPVals[kNotchAQ] = kDSPNotchAQ;
        m_Raw2DSPVals[kNotchACut] = kDSPNotchACut;

        ResetAll();
    }

```

0021



```
CDSP56kManager::~CDSP56kManager()
{
```

```
    if( m_Comm )
        delete m_Comm;
}
```

```
void CDSP56kManager::SetMainVolume(long value)
{
```

```
    double fvalue;
```

```
    if( value >= 0 )
        SetRawValue(value,kMainVolume);
    if( m_RawVals[kMainVolume] == DSP_CONTROL_MAX )
        m_DSPVals[kDSPMainVolume] = -m_RawVals[kMainVolume];
    else if( m_RawVals[kMainVolume] == 0 )
        m_DSPVals[kDSPMainVolume] = 0;
    else
    {
        fvalue = m_RawVals[kMainVolume];
        fvalue /= DSP_CONTROL_MAX;    // normalize
        fvalue *= VOLRANGE;
        fvalue += VOLBOT;
        fvalue = (pow(10.,fvalue) - 1.)/(10.-1.);
        fvalue *= DSP_CONTROL_MAX;
        m_DSPVals[kDSPMainVolume] = (long) -fvalue;
    }
    SendDSPValue(kDSPMainVolume);
}
```

```
void CDSP56kManager::SendDSPValue(long which)
{
```

```
    long value = m_DSPVals[which];
```

```
    #if 0
```

```
        // transmitted value has parameter value in top 16 bits and parameter identifier in 8 LSBs
        value = (value & 0xFFFF00) | (which & 0xFF);
        // m_DSPReturn = spi_xchg24(value);
        m_DSPReturn = m_Comm->SendDSPWord(value);
    #endif
```

```
    // changed protocol around to send index followed by data
    char buffer[6];
```

```
    buffer[0] = (which >> 16) & 0xFF;
    buffer[1] = (which >> 8) & 0xFF;
    buffer[2] = which & 0xFF;
    buffer[3] = (value >> 16) & 0xFF;
    buffer[4] = (value >> 8) & 0xFF;
    buffer[5] = value & 0xFF;
    m_DSPReturn = m_Comm->SendDSPMemory(buffer,6);
}
```

```
void CDSP56kManager::SetNotchFreq(long ivalue,long which)
{
```

```
    double value;
    int dspindex = m_Raw2DSPVals[which];
```

```
    if( ivalue >= 0 )
        SetRawValue(ivalue,which);
    value = ConvertFreqRange(which);
    value = - cos(m_pi*value);
    value *= DSP_CONTROL_MAX;
    m_DSPVals[dspindex] = (long) value;
```

0022

```

    SendDSPValue(dspindex);
    SetNotchQ(-1L, which+1, which);
}

#ifdef LEGACY
void CDSP56kManager::SetShelvFreq(long ivalue, long which)
{
    double value, wc;
    int dspindex = m_Raw2DSPVals[which];

    if( ivalue >= 0 )
        SetRawValue(ivalue, which);
    value = ConvertFreqRange(which);
    wc = m_pi*value;
    value = (tan(wc/2)-1)/(tan(wc/2)+1);
    value *= DSP_CONTROL_MAX;
    m_DSPVals[dspindex] = (long) value;
    SendDSPValue(dspindex);
}

#endif

void CDSP56kManager::SetNotchQ(long ivalue, long which, long rawfreq)
{
    double beta, tbeta, q, wc;
    int dspindex = m_Raw2DSPVals[which];

    if( ivalue >= 0 )
        SetRawValue(ivalue, which);
    wc = ConvertFreqRange(rawfreq);
    wc = m_pi*wc;
    q = ConvertQRange(m_RawVals[which]);
    tbeta = wc/(2*q);
    if( tbeta > m_pi/4 )
        tbeta = m_pi/4;
    beta = (1.-tan(tbeta))/(1.+tan(tbeta));
    beta *= DSP_CONTROL_MAX;
    m_DSPVals[dspindex] = (long) beta;
    SendDSPValue(dspindex);
}

void CDSP56kManager::SetBoostCut(long ivalue, long which)
{
    double value;
    int dspindex = m_Raw2DSPVals[which];

    if( ivalue >= 0 )
        SetRawValue(ivalue, which);
    value = ConvertBoostCutRange(m_RawVals[which]);
    value = pow(10., value);
    value /= GAIN_SCALE;
    value *= DSP_CONTROL_MAX;
    m_DSPVals[dspindex] = (long) value;
    SendDSPValue(dspindex);
}

void CDSP56kManager::SetShelv1Freq(long ivalue)
{
    // SetShelvFreq(value, kLoShelfFreq);
    double value, wc;

    if( ivalue >= 0 )
        SetRawValue(ivalue, kLoShelfFreq);
    value = m_RawVals[kLoShelfFreq];
    value /= DSP_CONTROL_MAX; // normalize

```

```

    value *= FLCRANGE;
    value += FLCBOTRANGE;
    value = (pow(10.,value) - 1.)/(10.-1.);
    wc = m_pi*value;
    value = (tan(wc/2)-1)/(tan(wc/2)+1);
    value *= DSP_CONTROL_MAX;
    m_DSPVals[kLoShelfFreq] = (long) value;
    SendDSPValue(kLoShelfFreq);
}

void CDSP56kManager::SetShelv1Boost(long value)
{
    SetBoostCut(value,kLoShelfGain);
}

void CDSP56kManager::SetShelv2Freq(long ivalue)
{
    // SetShelvFreq(value,kHiShelfFreq);
    double value,wc;

    if( ivalue >= 0 )
        SetRawValue(ivalue,kHiShelfFreq);
    value = m_RawVals[kHiShelfFreq];
    value /= DSP_CONTROL_MAX;    // normalize
    value *= FHCRANGE2;
    value += FHCBOTRANGE2;
    value = (pow(10.,value) - 1.)/(10.-1.);
    wc = m_pi*value;
    value = (tan(wc/2)-1)/(tan(wc/2)+1);
    value *= DSP_CONTROL_MAX;
    m_DSPVals[kHiShelfFreq] = (long) value;
    SendDSPValue(kHiShelfFreq);
}

void CDSP56kManager::SetShelv2Boost(long value)
{
    SetBoostCut(value,kHiShelfGain);
}

void CDSP56kManager::SetNotch1Freq(long value)
{
    SetNotchFreq(value,kNotch1Freq);
}

void CDSP56kManager::SetNotch1Q(long value)
{
    SetNotchQ(value,kNotch1Q,kNotch1Freq);
}

void CDSP56kManager::SetNotch1Cut(long value)
{
    SetBoostCut(value,kNotch1Cut);
}

void CDSP56kManager::SetNotch2Freq(long value)
{
    SetNotchFreq(value,kNotch2Freq);
}

```

```

}

void CDSP56kManager::SetNotch2Q(long value)
{
    SetNotchQ(value, kNotch2Q, kNotch2Freq);
}

void CDSP56kManager::SetNotch2Cut(long value)
{
    SetBoostCut(value, kNotch2Cut);
}

void CDSP56kManager::SetNotch9Freq(long value)
{
    double q, f, nf;

    SetNotchFreq(value, kNotch9Freq);
    f = ConvertFreqRange(kNotch9Freq);
    q = ConvertQRange(m_RawVals[kNotch9Q]);
    nf = f - (f / (5.*q));
    if( nf < 0. )
        nf = 0.;
    nf = pow(nf, 0.5);
    nf = log10((nf * (10.-1.)) + 1.);
    nf -= FBOTRANGE;
    nf /= (FTOPRANGE-FBOTRANGE);
    nf *= DSP_CONTROL_MAX;    // normalize
    m_RawVals[kNotch8Freq] = (long) nf;
    nf = f + (f / (5.*q));
    if( nf > 1. )
        nf = 1.;
    nf = pow(nf, 0.5);
    nf = log10((nf * (10.-1.)) + 1.);
    nf -= FBOTRANGE;
    nf /= (FTOPRANGE-FBOTRANGE);
    nf *= DSP_CONTROL_MAX;    // normalize
    m_RawVals[kNotchAFreq] = (long) nf;
    SetNotchFreq(-1L, kNotch8Freq);
    SetNotchFreq(-1L, kNotchAFreq);
}

void CDSP56kManager::SetNotch3Freq(long value)
{
    SetNotchFreq(value, kNotch3Freq);
    m_RawVals[kNotch5Freq] = (m_RawVals[kNotch4Freq] + m_RawVals[kNotch3Freq])/2;
    SetNotchFreq(-1L, kNotch5Freq);
}

void CDSP56kManager::SetNotch3Q(long value)
{
    SetNotchQ(value, kNotch3Q, kNotch3Freq);
}

void CDSP56kManager::SetNotch3Cut(long value)
{
    SetBoostCut(value, kNotch3Cut);
}

```

```

void CDSP56kManager::SetNotch4Freq(long value)
{
    SetNotchFreq(value, kNotch4Freq);
    m_RawVals[kNotch5Freq] = (m_RawVals[kNotch4Freq] + m_RawVals[kNotch3Freq]) / 2;
    SetNotchFreq(-1L, kNotch5Freq);
}

void CDSP56kManager::SetNotch4Q(long value)
{
    SetNotchQ(value, kNotch4Q, kNotch4Freq);
}

void CDSP56kManager::SetNotch4Cut(long value)
{
    SetBoostCut(value, kNotch4Cut);
}

void CDSP56kManager::SetNotch5Q(long value)
{
    SetNotchQ(value, kNotch5Q, kNotch5Freq);
}

void CDSP56kManager::SetNotch5Boost(long value)
{
    SetBoostCut(value, kNotch5Cut);
}

double CDSP56kManager::DoConvertFreqRange(double val, double top, double bottom)
{
    val /= DSP_CONTROL_MAX;    // normalize
    val *= top - bottom;
    val += bottom;
    val = (pow(10., val) - 1.) / (10. - 1.);
    return( val );
}

double CDSP56kManager::ConvertFreqRange(int which)
{
    double fval;

    switch( which )
    {
        case kNotch1Freq:
        case kNotch2Freq:
            fval = DoConvertFreqRange(m_RawVals[which], FTOPRANGE3, FBOTRANGE3);    // top frequency is 0.1 * Nyquis
            fval /= SAMPLING_FREQ/2;
            break;
        default:
            fval = DoConvertFreqRange(m_RawVals[which], FTOPRANGE, FBOTRANGE);
            // square log scale
            fval = fval * fval;
            break;
    }
    return( fval );
}

```

0026

```

// converts Q range to log scale
#define QTOPRANGE 2.258
#define QBOTRANGE 0.037
#define QRANGE (QTOPRANGE-QBOTRANGE)

double CDSP56kManager::ConvertQRange(long ival)
{
    double val;

    val = ival;
    val /= DSP_CONTROL_MAX;    // normalize
    val *= QRANGE;
    val += QBOTRANGE;
    val = (pow(10.,val) - 1.)/(10.-1.);
    return( val );
}

// converts input value to log gain value
// #define LOG8 0.9031
#define LOGGAINSCALE (log10(GAIN_SCALE))
double CDSP56kManager::ConvertBoostCutRange(long ival)
{
    double val;

    val = ival;
    val /= DSP_CONTROL_MAX;    // normalize
    val = (2.*val*LOGGAINSCALE) - LOGGAINSCALE;
    return( val );
}

void CDSP56kManager::SetRawValue(long value, long which)
{
    if( value )
        value += 1;
    value <= CONTROL_RANGE_SC;
    m_RawVals[which] = value;
}

long CDSP56kManager::GetParamValue(long which)
{
    long value;

    if( which >= kUIArraySize )
        value = 0x400000L >> CONTROL_RANGE_SC;
    else
        value = m_RawVals[which] >> CONTROL_RANGE_SC;
    value--;
    if( value < 0 )
        value = 0;
    return( value );
}

void CDSP56kManager::SetParamValue(long value, long which)
{
    switch( which )
    {
        case kHiCutoff2Freq:
            SetHiCutoff2Freq(value);
            break;
        case kHiCutoff2Q:
            SetHiCutoff2Q(value);
            break;
        case kHiCutoffFreq:

```

0027

```

    SetHiCutoffFreq(value);
    break;
case kHiCutoffQ:
    SetHiCutoffQ(value);
    break;
case kLoCutoffFreq:
    SetLoCutoffFreq(value);
    break;
case kLoCutoffQ:
    SetLoCutoffQ(value);
    break;
case kBypass:
    SetBypass(value);
    break;
case kDDXCompBypass:
    SetDDXCompBypass(value);
    break;
case kHDCDBypass:
    SetHDCDBypass(value);
    break;
case kHDCDGainScaleOff:
    SetHDCDGainScale(value);
    break;
case kAnalogVolume:
    SetAnalogVolume(value);
    break;
case kMainVolume:
    SetMainVolume(value);
    break;
case kPreVolume:
    SetPreVolume(value);
    break;
case kLoShelfFreq:
    SetShelv1Freq(value);
    break;
case kHiShelfFreq:
    SetShelv2Freq(value);
    break;
case kNotch8Cut:
    SetBoostCut(value, which);
    SetBoostCut(value, kNotchACut); // same value for both notches
    break;
case kNotch1Cut:
case kNotch2Cut:
case kNotch3Cut:
case kNotch4Cut:
case kNotch5Cut:
case kNotch6Cut:
case kNotch7Cut:
case kLoShelfGain:
case kHiShelfGain:
    SetBoostCut(value, which);
    break;
case kNotch9Cut:
    m_DSPVals[kDSPNotch9Cut] = 0; // fixed notch
    SendDSPValue(kDSPNotch9Cut);
    break;
case kNotch3Freq:
    SetNotch3Freq(value);
    break;
case kNotch4Freq:
    SetNotch4Freq(value);
    break;
case kNotch9Freq:
    SetNotch9Freq(value);
    break;
case kNotch1Freq:
case kNotch2Freq:
case kNotch5Freq:
case kNotch6Freq:
case kNotch7Freq:
case kAllpassFreq:

```

0028

```

        SetNotchFreq(value, which);
        break;
    case kNotch8Q:
        SetNotchQ(value, which, which-1);
        SetNotchQ(value, kNotchAQ, kNotchAQ-1);    // same value for both notches
        break;
    case kNotch9Q:
        SetNotchQ(value, which, which-1);
        SetNotch9Freq(-1);
        break;
    case kNotch1Q:
    case kNotch2Q:
    case kNotch3Q:
    case kNotch4Q:
    case kNotch5Q:
    case kNotch6Q:
    case kNotch7Q:
    case kAllpassQ:
        SetNotchQ(value, which, which-1);
        break;
    case kDelay:
        SetDelay(value);
        break;
    // case kDBNotchWidth:
    // SetDelay(value);
    // break;
    case kNotch8Freq:
    case kNotchAFreq:
    case kNotchACut:
    case kNotchAQ:
        break;
    }
}

```

```

void CDSP56kManager::SetDSPSettings(CStringArray & array)
{

```

```

    int i;
    CString string;
    int index;
    long value;
    int fileversion = 0;
    int j = 0;
    double fvalue;
    const char *sptr;

```

```

    // reset all parameters
    ResetAll();

```

```

    // first check for comment at head
    string = array.GetAt(0);
    sptr = string;
    if( strcmp(sptr, "# VERSION") == 0 )
    {

```

```

        string = array.GetAt(1);
        sptr = string;
        sscanf(sptr, "%d", &fileversion);
        j = 2;
    }

```

```

    for( i = j ; i < array.GetSize() ; i++ )
    {

```

```

        string = array.GetAt(i);
        sptr = string;
        if( sptr[0] == '#' ) continue;    // skip comments
        sscanf(sptr, "%d\t0x%x", &index, &value);
        if( index >= kUIArraySize ) continue;
        // Add any special treatment of parameters here
        if( fileversion == 0L )
        {
            switch( index )

```

0029



```

{
case kNotch1Freq:
case kNotch2Freq:
    fvalue = DoConvertFreqRange(value, FTOPRANGE, FBOTRANGE);
    fvalue = fvalue * fvalue;
    fvalue *= SAMPLING_FREQ/2;
    fvalue = log10((fvalue * (10.-1.)) + 1.);
    fvalue -= FBOTRANGE3;
    fvalue /= (FTOPRANGE3-FBOTRANGE3);
    if( fvalue > 1.0 )
        fvalue = 1.0;
    fvalue *= DSP_CONTROL_MAX;    // normalize
    value = (long) fvalue;
    break;
    // increased low frequency
case kHiShelfFreq:
    fvalue = DoConvertFreqRange(value, FHCTOPRANGE, FHCBOTRANGE);
    fvalue = log10((fvalue * (10.-1.)) + 1.);
    fvalue -= FHCBOTRANGE2;
    fvalue /= (FHCTOPRANGE-FHCBOTRANGE2);
    if( fvalue > 1.0 )
        fvalue = 1.0;
    fvalue *= DSP_CONTROL_MAX;    // normalize
    value = (long) fvalue;
    break;
case kHiCutoffFreq:
    fvalue = DoConvertFreqRange(value, FHCTOPRANGE, FHCBOTRANGE);
    fvalue *= SAMPLING_FREQ/2;
    fvalue = log10((fvalue * (10.-1.)) + 1.);
    fvalue -= FHCBOTRANGE3;
    fvalue /= (FHCTOPRANGE3-FHCBOTRANGE3);
    if( fvalue > 1.0 )
        fvalue = 1.0;
    fvalue *= DSP_CONTROL_MAX;    // normalize
    value = (long) fvalue;
    break;
}
}
else if( fileversion == 1L )
{
    switch( index )
    {
    case kNotch1Freq:
    case kNotch2Freq:
        fvalue = DoConvertFreqRange(value, FTOPRANGE2, FBOTRANGE2);
        fvalue *= SAMPLING_FREQ/2;
        fvalue = log10((fvalue * (10.-1.)) + 1.);
        fvalue -= FBOTRANGE3;
        fvalue /= (FTOPRANGE3-FBOTRANGE3);
        if( fvalue > 1.0 )
            fvalue = 1.0;
        fvalue *= DSP_CONTROL_MAX;    // normalize
        value = (long) fvalue;
        break;
    case kLoCutoffFreq:
        fvalue = DoConvertFreqRange(value, FLCTOPRANGE, FLCBOTRANGE);
        fvalue *= SAMPLING_FREQ/2;
        fvalue = log10((fvalue * (10.-1.)) + 1.);
        fvalue -= FLCBOTRANGE2;
        fvalue /= (FLCTOPRANGE2-FLCBOTRANGE2);
        if( fvalue > 1.0 )
            fvalue = 1.0;
        fvalue *= DSP_CONTROL_MAX;    // normalize
        value = (long) fvalue;
        break;
    case kHiCutoff2Freq:
        fvalue = DoConvertFreqRange(value, FTOPRANGE, FBOTRANGE);
        fvalue *= SAMPLING_FREQ/2;
        fvalue = log10((fvalue * (10.-1.)) + 1.);
        fvalue -= FHC2BOT;
        fvalue /= (FHC2TOP-FHC2BOT);
        if( fvalue > 1.0 )

```

0030

```

        fvalue = 1;
        fvalue *= DSP_CONTROL_MAX;    // normalize
        value = (long) fvalue;
        break;
    case kHiCutoffFreq:
        fvalue = DoConvertFreqRange(value, FHCTOPRANGE, FHCBOTRANGE2);
        fvalue *= SAMPLING_FREQ/2;
        fvalue = log10((fvalue * (10.-1.)) + 1.);
        fvalue -= FHCBOTRANGE3;
        fvalue /= (FHCTOPRANGE3-FHCBOTRANGE3);
        if( fvalue > 1.0 )
            fvalue = 1.0;
        fvalue *= DSP_CONTROL_MAX;    // normalize
        value = (long) fvalue;
        break;
    }
    m_RawVals[index] = value;
}
// tell DSP of changes
for( i = 0 ; i < kUIArraySize ; i++ )
{
    SetParamValue(-1,i);
}
m_DSPVals[kDSPBypassMask] = m_RawVals[kBypassMask];
SendDSPValue(kDSPBypassMask);
}

```

```

void CDSP56kManager::GetDSPSettings(CStringArray & array)
{

```

```

    char s[1000];
    int i,index;
    int j = 0;
    CString cstr;

    // write header (so far only version number)
    cstr = "# VERSION";
    array.SetAtGrow(j++, cstr);
    sprintf(s,"%d",FILE_VERSION);
    array.SetAtGrow(j++, s);
    cstr = "# DATA";
    array.SetAtGrow(j++, cstr);

    for( i = 0 ; i < kUIArraySize ; i++ )
    {
        index = kUIArraySize-i-1;
        if( index == kBypass ) continue;
        if( index == kHDCDBypass ) continue;
        if( index == kHDCDGainScaleOff ) continue;
        if( index == kAnalogIn ) continue;
        // MM 7/14/99 Added value in human readable form as comment
        GetStringValue(index,cstr);
        sprintf(s,"# %s %s",m_ParamNames[index],cstr);
        array.SetAtGrow(j++, s);
        sprintf(s,"%d\t0x%x",index,m_RawVals[index]);
        array.SetAtGrow(j++, s);
    }
}

```

```

void CDSP56kManager::GetFilterBlob(CStringArray & array)
{

```

```

    char s[1000];
    int param;
    int j = 0;
    CString cstr;
    float v;

    // write version number

```

0031

```

// cstr = "# VERSION";
// array.SetAtGrow(j++, cstr);
sprintf(s, "%d", BLOB_FILE_VERSION);
array.SetAtGrow(j++, s);

// output delay
param = kDelay;
GetPureStringValue(param, cstr);
sscanf(cstr, "%f", &v);
if( v != 0. )
{
// cstr = m_ParamNames[param];
// array.SetAtGrow(j++, cstr);
sprintf(s, "%d", BLOCK_DELAY);
array.SetAtGrow(j++, s);
cstr = "1";
array.SetAtGrow(j++, cstr);
GetPureStringValue(param, cstr);
array.SetAtGrow(j++, cstr);
}

// output pre-gain
param = kPreVolume;
// cstr = m_ParamNames[param];
// array.SetAtGrow(j++, cstr);
sprintf(s, "%d", BLOCK_GAIN);
array.SetAtGrow(j++, s);
cstr = "1";
array.SetAtGrow(j++, cstr);
GetPureStringValue(param, cstr);
array.SetAtGrow(j++, cstr);

// lo-shelf
if( !GetBypassSection(kBypassLoshelf) )
{
// cstr = "Low-Shelf";
// array.SetAtGrow(j++, cstr);
sprintf(s, "%d", BLOCK_LS);
array.SetAtGrow(j++, s);
cstr = "2";
array.SetAtGrow(j++, cstr);
GetPureStringValue(kLoShelfFreq, cstr);
array.SetAtGrow(j++, cstr);
GetPureStringValue(kLoShelfGain, cstr);
array.SetAtGrow(j++, cstr);
}

// hi-shelf
if( !GetBypassSection(kBypassHiShelf) )
{
// cstr = "High-Shelf";
// array.SetAtGrow(j++, cstr);
sprintf(s, "%d", BLOCK_HS);
array.SetAtGrow(j++, s);
cstr = "2";
array.SetAtGrow(j++, cstr);
GetPureStringValue(kHiShelfFreq, cstr);
array.SetAtGrow(j++, cstr);
GetPureStringValue(kHiShelfGain, cstr);
array.SetAtGrow(j++, cstr);
}

// low peaking cutoff
if( !GetBypassSection(kBypassHipass) )
{
// cstr = "Low Cutoff";
// array.SetAtGrow(j++, cstr);
sprintf(s, "%d", BLOCK_HP);
array.SetAtGrow(j++, s);
cstr = "2";
array.SetAtGrow(j++, cstr);
GetPureStringValue(kLoCutoffFreq, cstr);

```

0032

```

    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kLoCutoffQ, cstr);
    array.SetAtGrow(j++, cstr);
}

// high peaking cutoff #1
if( !GetBypassSection(kBypassLopass) )
{
    // cstr = "High Cutoff";
    // array.SetAtGrow(j++, cstr);
    sprintf(s, "%d", BLOCK_LP);
    array.SetAtGrow(j++, s);
    cstr = "2";
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kHiCutoffFreq, cstr);
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kHiCutoffQ, cstr);
    array.SetAtGrow(j++, cstr);
}

// high peaking cutoff #2
if( !GetBypassSection(kBypassNlopass) )
{
    // cstr = "High Cutoff #2";
    // array.SetAtGrow(j++, cstr);
    sprintf(s, "%d", BLOCK_LP2);
    array.SetAtGrow(j++, s);
    cstr = "2";
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kHiCutoff2Freq, cstr);
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kHiCutoff2Q, cstr);
    array.SetAtGrow(j++, cstr);
}

// resonance comp #1
if( !GetBypassSection(kBypassNotch1) )
{
    // cstr = "Parametric EQ";
    // array.SetAtGrow(j++, cstr);
    sprintf(s, "%d", BLOCK_PEQ);
    array.SetAtGrow(j++, s);
    cstr = "3";
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kNotch1Freq, cstr);
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kNotch1Q, cstr);
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kNotch1Cut, cstr);
    array.SetAtGrow(j++, cstr);
}

// resonance comp #2
if( !GetBypassSection(kBypassNotch2) )
{
    // cstr = "Parametric EQ";
    // array.SetAtGrow(j++, cstr);
    sprintf(s, "%d", BLOCK_PEQ);
    array.SetAtGrow(j++, s);
    cstr = "3";
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kNotch2Freq, cstr);
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kNotch2Q, cstr);
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(kNotch2Cut, cstr);
    array.SetAtGrow(j++, cstr);
}

// resonance comp #3
if( !GetBypassSection(kBypassNotch3) )
{

```

0033

```

//      cstr = "Parametric EQ";
//      array.SetAtGrow(j++, cstr);
      sprintf(s, "%d", BLOCK_PEQ);
      array.SetAtGrow(j++, s);
      cstr = "3";
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch6Freq, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch6Q, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch6Cut, cstr);
      array.SetAtGrow(j++, cstr);
}

// resonance comp #4
if( !GetBypassSection(kBypassNotch4) )
{
//      cstr = "Parametric EQ";
//      array.SetAtGrow(j++, cstr);
      sprintf(s, "%d", BLOCK_PEQ);
      array.SetAtGrow(j++, s);
      cstr = "3";
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch7Freq, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch7Q, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch7Cut, cstr);
      array.SetAtGrow(j++, cstr);
}

// conecry
if( !GetBypassSection(kBypassConecry) )
{
//      cstr = "Parametric EQ";
//      array.SetAtGrow(j++, cstr);
      sprintf(s, "%d", BLOCK_PEQ);
      array.SetAtGrow(j++, s);
      cstr = "3";
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch3Freq, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch3Q, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch3Cut, cstr);
      array.SetAtGrow(j++, cstr);
}

//      cstr = "Parametric EQ";
//      array.SetAtGrow(j++, cstr);
      sprintf(s, "%d", BLOCK_PEQ);
      array.SetAtGrow(j++, s);
      cstr = "3";
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch4Freq, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch4Q, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch4Cut, cstr);
      array.SetAtGrow(j++, cstr);

//      cstr = "Parametric EQ";
//      array.SetAtGrow(j++, cstr);
      sprintf(s, "%d", BLOCK_PEQ);
      array.SetAtGrow(j++, s);
      cstr = "3";
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch5Freq, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch5Q, cstr);
      array.SetAtGrow(j++, cstr);
      GetPureStringValue(kNotch5Cut, cstr);
      array.SetAtGrow(j++, cstr);

```

0034

```

    }

    // allpass
    if( !GetBypassSection(kBypassAllpass) )
    {
        // cstr = "Parametric EQ";
        // array.SetAtGrow(j++, cstr);
        sprintf(s,"%d",BLOCK_AP);
        array.SetAtGrow(j++, s);
        cstr = "2";
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kAllpassFreq,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kAllpassQ,cstr);
        array.SetAtGrow(j++, cstr);
    }

    // double-tuned notch
    if( !GetBypassSection(kBypassDBNotch) )
    {
        // cstr = "Parametric EQ";
        // array.SetAtGrow(j++, cstr);
        sprintf(s,"%d",BLOCK_PEQ);
        array.SetAtGrow(j++, s);
        cstr = "3";
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotch8Freq,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotch8Q,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotch8Cut,cstr);
        array.SetAtGrow(j++, cstr);

        // cstr = "Parametric EQ";
        // array.SetAtGrow(j++, cstr);
        sprintf(s,"%d",BLOCK_PEQ);
        array.SetAtGrow(j++, s);
        cstr = "3";
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotch9Freq,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotch9Q,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotch9Cut,cstr);
        array.SetAtGrow(j++, cstr);

        // cstr = "Parametric EQ";
        // array.SetAtGrow(j++, cstr);
        sprintf(s,"%d",BLOCK_PEQ);
        array.SetAtGrow(j++, s);
        cstr = "3";
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotchAFreq,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotchAQ,cstr);
        array.SetAtGrow(j++, cstr);
        GetPureStringValue(kNotchACut,cstr);
        array.SetAtGrow(j++, cstr);
    }

    // output post-gain
    param = kMainVolume;
    // cstr = m_ParamNames[param];
    // array.SetAtGrow(j++, cstr);
    sprintf(s,"%d",BLOCK_GAIN);
    array.SetAtGrow(j++, s);
    cstr = "1";
    array.SetAtGrow(j++, cstr);
    GetPureStringValue(param,cstr);
    array.SetAtGrow(j++, cstr);

```

```

}

void CDSP56kManager::SetPreVolume(long value)
{
    double fvalue;

    if( value >= 0 )
        SetRawValue(value,kPreVolume);
    if( m_RawVals[kPreVolume] == DSP_CONTROL_MAX )
        m_DSPVals[kDSPPreVolume] = -m_RawVals[kPreVolume];
    else if( m_RawVals[kPreVolume] == 0 )
        m_DSPVals[kDSPPreVolume] = 0;
    else
    {
        fvalue = m_RawVals[kPreVolume];
        fvalue /= DSP_CONTROL_MAX;    // normalize
        fvalue *= VOLRANGE;
        fvalue += VOLBOT;
        fvalue = (pow(10.,fvalue) - 1.)/(10.-1.);
        fvalue *= DSP_CONTROL_MAX;
        m_DSPVals[kDSPPreVolume] = (long) -fvalue;
    }
    SendDSPValue(kDSPPreVolume);
}

void CDSP56kManager::SetAnalogVolume(long value)
{
    double fvalue;

    if( value >= 0 )
        SetRawValue(value,kAnalogVolume);
    if( m_RawVals[kAnalogVolume] == DSP_CONTROL_MAX )
        m_DSPVals[kDSPAnalogVol] = -m_RawVals[kAnalogVolume];
    else if( m_RawVals[kAnalogVolume] == 0 )
        m_DSPVals[kDSPAnalogVol] = 0;
    else
    {
        fvalue = m_RawVals[kAnalogVolume];
        fvalue /= DSP_CONTROL_MAX;    // normalize
        fvalue *= VOLRANGE;
        fvalue += VOLBOT;
        fvalue = (pow(10.,fvalue) - 1.)/(10.-1.);
        fvalue *= DSP_CONTROL_MAX;
        m_DSPVals[kDSPAnalogVol] = (long) -fvalue;
    }
    SendDSPValue(kDSPAnalogVol);
}

void CDSP56kManager::SetBypass(long value)
{
    if( value >= 0 )
        SetRawValue(value,kBypass);
    if( m_RawVals[kBypass] )
        m_DSPVals[kDSPBypass] = 0x7FFFFFFF;
    else
        m_DSPVals[kDSPBypass] = 0;
    SendDSPValue(kDSPBypass);
}

void CDSP56kManager::SetLoCutoffFreq(long value)
{
    if( value >= 0 )
        SetRawValue(value,kLoCutoffFreq);

```

0036

```

    SetLoCutoff();
}

void CDSP56kManager::SetLoCutoffQ(long value)
{
    if( value >= 0 )
        SetRawValue(value, kLoCutoffQ);
    SetLoCutoff();
}

void CDSP56kManager::SetHiCutoffFreq(long value)
{
    if( value >= 0 )
        SetRawValue(value, kHiCutoffFreq);
    SetHiCutoff();
}

void CDSP56kManager::SetHiCutoffQ(long value)
{
    if( value >= 0 )
        SetRawValue(value, kHiCutoffQ);
    SetHiCutoff();
}

void CDSP56kManager::SetHiCutoff2Freq(long value)
{
    if( value >= 0 )
        SetRawValue(value, kHiCutoff2Freq);
    SetHiCutoff2();
}

void CDSP56kManager::SetHiCutoff2Q(long value)
{
    if( value >= 0 )
        SetRawValue(value, kHiCutoff2Q);
    SetHiCutoff2();
}

void CDSP56kManager::SetHiCutoff()
{
    double f,q;
    double a1,a2,b0;

    q = m_RawVals[kHiCutoffQ];
    q /= DSP_CONTROL_MAX; // normalize
    q *= QLCRANGE;
    q += QLCBOTRANGE;
    q = (pow(10.,q) - 1.)/(10.-1.);
    if( q >= 1. )
        q = 0.99999;

    /*f = m_RawVals[kHiCutoffFreq];
    f /= DSP_CONTROL_MAX; // normalize
    f *= FHCRANGE2;
    f += FHCBOTRANGE2;
    f = (pow(10.,f) - 1.)/(10.-1.); */
    f = DoConvertFreqRange(m_RawVals[kHiCutoffFreq], FHCTOPRANGE3, FHCBOTRANGE3);
    f /= SAMPLING_FREQ/2;
    f = 2. * sin( f * m_pi / 2.);

```



```

a2 = 1. - f * q;
if( a2 >= 2. )
    a2 = 1.99999;
else if( a2 <= -2. )
    a2 = -1.99999;
a1 = -(2. - f * q - f * f);
if( a1 >= 2. )
    a1 = 1.99999;
else if( a1 <= -2. )
    a1 = -1.99999;
b0 = f * f / 2.;
if( b0 >= 8. )
    b0 = 7.99999;
else if( b0 <= -8. )
    b0 = -7.99999;

m_DSPVals[kDSPHiCutoffA2] = (long) (a2 * DSP_CONTROL_MAX/ 2.);
SendDSPValue(kDSPHiCutoffA2);
m_DSPVals[kDSPHiCutoffA1] = (long) (a1 * DSP_CONTROL_MAX/2.);
SendDSPValue(kDSPHiCutoffA1);
m_DSPVals[kDSPHiCutoffScale] = (long) (b0 * DSP_CONTROL_MAX/ 16.);
SendDSPValue(kDSPHiCutoffScale);
}

void CDSP56kManager::SetHiCutoff2()
{
    double f,q;
    double a1,a2,b0;
    double gamma,beta,lambda,alpha;

    q = DSP_CONTROL_MAX - m_RawVals[kHiCutoff2Q]; // invert scale
    q /= DSP_CONTROL_MAX; // normalize
    q *= (QHC2TOP-QHC2BOT);
    q += QHC2BOT;
    q = (pow(10.,q) - 1.)/(10.-1.);
    // f = ConvertFreqRange(kHiCutoff2Freq);
    f = DoConvertFreqRange(m_RawVals[kHiCutoff2Freq], FHC2TOP, FHC2BOT);
    f /= SAMPLING_FREQ/2;

    gamma = - cos(f * m_pi);
    beta = (1. - sin(f*m_pi/(2*q)))/(1. + sin(f*m_pi/(2*q)));
    if( beta > 2.0 )
        beta = 2.0;
    lambda = 4. * beta * (gamma/(1.+beta));
    if( lambda > 2.0 )
        lambda = 2.0;
    alpha = 1. + beta + lambda;

    a2 = beta;
    a1 = lambda;
    b0 = alpha / 4.;
    if( b0 >= 8. )
        b0 = 7.99999;
    else if( b0 <= -8. )
        b0 = -7.99999;

    m_DSPVals[kDSP2HiCutoffA2] = (long) (a2 * DSP_CONTROL_MAX/ 2.);
    SendDSPValue(kDSP2HiCutoffA2);
    m_DSPVals[kDSP2HiCutoffA1] = (long) (a1 * DSP_CONTROL_MAX/2.);
    SendDSPValue(kDSP2HiCutoffA1);
    m_DSPVals[kDSP2HiCutoffScale] = (long) (b0 * DSP_CONTROL_MAX/ 16.);
    SendDSPValue(kDSP2HiCutoffScale);
}

void CDSP56kManager::SetLoCutoff()
{
    double f,q,scale_factor;

```

```

q = m_RawVals[kLoCutoffQ];
q /= DSP_CONTROL_MAX; // normalize
q *= QLCRANGE;
q += QLCBOTRANGE;
q = (pow(10.,q) - 1.)/(10.-1.);
if( q >= 1. )
    q = 0.99999;
f = m_RawVals[kLoCutoffFreq];
f /= DSP_CONTROL_MAX; // normalize
f *= (FLCTOPRANGE2-FLCBOTRANGE2);
f += FLCBOTRANGE2;
f = (pow(10.,f) - 1.)/(10.-1.);
f /= SAMPLING_FREQ/2;
f = 2. * sin( f * m_pi / 2.);
if( f >= 1. )
    f = 0.99999;
scale_factor = (4. - 2. * f * q - f * f)/4.;

m_DSPVals[kDSPLoCutoffFc] = (long) (f * DSP_CONTROL_MAX);
SendDSPValue(kDSPLoCutoffFc);
m_DSPVals[kDSPLoCutoffQc] = (long) (q * DSP_CONTROL_MAX);
SendDSPValue(kDSPLoCutoffQc);
m_DSPVals[kDSPLoCutoffScale] = (long) (-0.5 * scale_factor * DSP_CONTROL_MAX);
SendDSPValue(kDSPLoCutoffScale);
}

void CDSP56kManager::SetDDXCompBypass(long value)
{
    if( value >= 0 )
        SetRawValue(value, kDDXCompBypass);
    if( m_RawVals[kDDXCompBypass] )
        m_DSPVals[kDSPDDXCompBypass] = 0x7FFFFFFF;
    else
        m_DSPVals[kDSPDDXCompBypass] = 0;
    SendDSPValue(kDSPDDXCompBypass);
}

void CDSP56kManager::SetHDCDBypass(long value)
{
    if( value >= 0 )
        SetRawValue(value, kHDCDBypass);
    if( m_RawVals[kHDCDBypass] )
        m_DSPVals[kDSPHDCDBypass] = 0x7FFFFFFF;
    else
        m_DSPVals[kDSPHDCDBypass] = 0;
    SendDSPValue(kDSPHDCDBypass);
}

void CDSP56kManager::SetHDCDGainScale(long value)
{
    if( value >= 0 )
        SetRawValue(value, kHDCDGainScaleOff);
    if( m_RawVals[kHDCDGainScaleOff] )
        m_DSPVals[kDSPHDCDGainScaleOff] = 0x7FFFFFFF;
    else
        m_DSPVals[kDSPHDCDGainScaleOff] = 0;
    SendDSPValue(kDSPHDCDGainScaleOff);
}

void CDSP56kManager::ResetAll()
{

```

```

int i;
for( i = 0 ; i < kUIArraySize ; i++ )
{
    m_RawVals[i] = 0x400000L;
}
for( i = 0 ; i < kDSPArraySize ; i++ )
{
    m_DSPVals[i] = 0;
}

m_RawVals[kAnalogVolume] = 0x800000L;
m_RawVals[kMainVolume] = 0x800000L;
m_RawVals[kPreVolume] = 0x800000L;
m_RawVals[kBypass] = 0x000000L;
m_RawVals[kHDCDBypass] = 0x000000L;
m_RawVals[kDDXCompBypass] = 0x000000L;
m_RawVals[kHDCDGainScaleOff] = 0x000000L;
m_RawVals[kAnalogIn] = 0x000000L;
m_RawVals[kDelay] = 0x000000L;

m_RawVals[kHiCutoff2Freq] = 0x600000L;
m_RawVals[kHiCutoff2Q] = 0x200000L;
m_RawVals[kHiCutoffFreq] = 0x600000L;
m_RawVals[kHiCutoffQ] = 0x200000L;
m_RawVals[kLoCutoffFreq] = 0x400000L;
m_RawVals[kLoCutoffQ] = 0x200000L;

m_DSPReturn = 0;

for( i = 0 ; i < kUIArraySize ; i++ )
{
    SetParamValue(-1,i);
}

m_DSPVals[kDSPBypassMask] = m_RawVals[kBypassMask] = 0L;
SendDSPValue(kDSPBypassMask);
}

void CDSP56kManager::SetNotch6Freq(long value)
{
    SetNotchFreq(value,kNotch6Freq);
}

void CDSP56kManager::SetNotch6Q(long value)
{
    SetNotchQ(value,kNotch6Q,kNotch6Freq);
}

void CDSP56kManager::SetNotch6Cut(long value)
{
    SetBoostCut(value,kNotch6Cut);
}

void CDSP56kManager::SetNotch7Freq(long value)
{
    SetNotchFreq(value,kNotch7Freq);
}

void CDSP56kManager::SetNotch7Q(long value)
{
    SetNotchQ(value,kNotch7Q,kNotch7Freq);
}

```

0040

```

}

void CDSP56kManager::SetNotch7Cut(long value)
{
    SetBoostCut(value, kNotch7Cut);
}

int CDSP56kManager::GetHDCDMode()
{
    // SetParamValue(-1, kMainVolume);
    // return( m_DSPReturn );
    return( m_Comm->GetDetectState() );
}

void CDSP56kManager::SetAnalogInput(long value)
{
    if( value >= 0 )
        SetRawValue(value, kAnalogIn);
    if( m_RawVals[kAnalogIn] )
        m_DSPVals[kDSPAnalogIn] = 0x7FFFFFFF;
    else
        m_DSPVals[kDSPAnalogIn] = 0;
    SendDSPValue(kDSPAnalogIn);
}

BOOL CDSP56kManager::IsReady()
{
    return( m_Comm->CheckState() );
}

void CDSP56kManager::GetPureStringValue(int which, CString &str)
{
    int count;

    GetStringValue(which, str);
    str.TrimLeft();
    count = str.Find(' ');
    if( count >= 0 )
        str = str.Left(count);
}

void CDSP56kManager::GetStringValue(int which, CString &str)
{
    double value;
    char s[100];

    switch( which )
    {
        case kAnalogVolume:
        case kMainVolume:
        case kPreVolume:
            value = m_RawVals[which];
            value /= DSP_CONTROL_MAX;
            value = 20 * log10(value);
            sprintf(s, "%10.2f dB", value);
            str = s;
            break;
        case kHiCutoffFreq:
            /* value = m_RawVals[which];
            value /= DSP_CONTROL_MAX; // normalize
            value *= FHCRange2;
            */
    }
}

```

0041

```

value += FHCBOTRANGE;
value = (pow(10.,value) - 1.)/(10.-1.);
value *= SAMPLING_FREQ/2;
value = DoConvertFreqRange(m_RawVals[kHiCutoffFreq], FHCTOPRANGE3, FHCBOTRANGE3);
sprintf(s, "%10.2f Hz", value);
str = s;
break;
case kHiCutoff2Q:
value = DSP_CONTROL_MAX - m_RawVals[kHiCutoff2Q];
value /= DSP_CONTROL_MAX; // normalize
value *= QHC2TOP-QHC2BOT;
value += QHC2BOT;
value = (pow(10.,value) - 1.)/(10.-1.);
sprintf(s, "%10.2f", value);
str = s;
break;
case kHiCutoffQ:
value = m_RawVals[kHiCutoffQ];
value /= DSP_CONTROL_MAX; // normalize
value *= QLCRANGE;
value += QLCBOTRANGE;
value = (pow(10.,value) - 1.)/(10.-1.);
if( value >= 1. )
value = 0.99999;
sprintf(s, "%10.2f", value);
str = s;
break;
case kLoCutoffFreq:
value = m_RawVals[kLoCutoffFreq];
value /= DSP_CONTROL_MAX; // normalize
value *= (FLCTOPRANGE2-FLCBOTRANGE2);
value += FLCBOTRANGE2;
value = (pow(10.,value) - 1.)/(10.-1.);
sprintf(s, "%10.2f Hz", value);
str = s;
break;
case kLoCutoffQ:
value = m_RawVals[kLoCutoffQ];
value /= DSP_CONTROL_MAX; // normalize
value *= QLCRANGE;
value += QLCBOTRANGE;
value = (pow(10.,value) - 1.)/(10.-1.);
if( value >= 1. )
value = 0.99999;
sprintf(s, "%10.2f", value);
str = s;
break;
case kLoShelfFreq:
value = m_RawVals[kLoShelfFreq];
value /= DSP_CONTROL_MAX; // normalize
value *= FLCRANGE;
value += FLCBOTRANGE;
value = (pow(10.,value) - 1.)/(10.-1.);
value *= SAMPLING_FREQ/2;
sprintf(s, "%10.2f Hz", value);
str = s;
break;
case kHiShelfFreq:
value = m_RawVals[kHiShelfFreq];
value /= DSP_CONTROL_MAX; // normalize
value *= FHCRANGE2;
value += FHCBOTRANGE2;
value = (pow(10.,value) - 1.)/(10.-1.);
value *= SAMPLING_FREQ/2;
sprintf(s, "%10.2f Hz", value);
str = s;
break;
case kNotch1Cut:
case kNotch2Cut:
case kNotch3Cut:
case kNotch4Cut:
case kNotch5Cut:

```

0042

```

case kNotch6Cut:
case kNotch7Cut:
case kNotch8Cut:
case kNotch9Cut:
case kNotchACut:
case kLoShelfGain:
case kHiShelfGain:
    value = 20 * ConvertBoostCutRange(m_RawVals[which]);
    sprintf(s,"%10.2f dB",value);
    str = s;
    break;
case kNotch1Freq:
case kNotch2Freq:
case kNotch3Freq:
case kNotch4Freq:
case kNotch5Freq:
case kNotch6Freq:
case kNotch7Freq:
case kNotch8Freq:
case kNotch9Freq:
case kNotchAFreq:
case kAllpassFreq:
    value = ConvertFreqRange(which);
    value *= SAMPLING_FREQ/2;
    sprintf(s,"%10.2f Hz",value);
    str = s;
    break;
case kHiCutoff2Freq:
    value = DoConvertFreqRange(m_RawVals[kHiCutoff2Freq],FHC2TOP,FHC2BOT);
    sprintf(s,"%10.2f Hz",value);
    str = s;
    break;
case kNotch1Q:
case kNotch2Q:
case kNotch3Q:
case kNotch4Q:
case kNotch5Q:
case kNotch6Q:
case kNotch7Q:
case kNotch8Q:
case kNotch9Q:
case kNotchAQ:
case kAllpassQ:
    value = ConvertQRange(m_RawVals[which]);
    sprintf(s,"%10.2f",value);
    str = s;
    break;
case kDelay:
    value = m_RawVals[which];
    value /= DSP_CONTROL_MAX;
    value *= DELAY_LENGTH;
    value /= SAMPLING_FREQ;
    value *= 1000.;
    sprintf(s,"%10.2f mS",value);
    str = s;
    break;
default:
    str = "fixed";
    break;
}
}

BOOL CDSP56kManager::IsBusy()
{
    return( m_Comm->IsTransmitting() );
}

void CDSP56kManager::SetBypassSection(BOOL value, int which)
{

```

```

long mask = 1L << which;

if( value )
{
    m_RawVals[kBypassMask] |= mask;
}
else
{
    mask = -mask;
    m_RawVals[kBypassMask] &= mask;
}
m_DSPVals[kDSPBypassMask] = m_RawVals[kBypassMask];
SendDSPValue(kDSPBypassMask);
}

BOOL CDSP56kManager::GetBypassSection(int which)
{
    long mask = 1L << which;

    return( (m_RawVals[kBypassMask] & mask) != 0 );
}

void CDSP56kManager::SetDelay(long value)
{
    if( value >= 0 )
    {
        SetRawValue(value,kDelay);
        m_DSPVals[kDSPDelay] = m_RawVals[kDelay];
        SendDSPValue(kDSPDelay);
    }
}

```

```

// UI IDs
enum {
    kMainVolume,
    kLoShelfFreq,
    kLoShelfGain,
    kHiShelfFreq,
    kHiShelfGain,
    kNotch1Freq,
    kNotch1Q,
    kNotch1Cut,
    kNotch2Freq,
    kNotch2Q,
    kNotch2Cut,
    kNotch3Freq,
    kNotch3Q,
    kNotch3Cut,
    kNotch4Freq,
    kNotch4Q,
    kNotch4Cut,
    kNotch5Freq,
    kNotch5Q,
    kNotch5Cut,
    kPreVolume,
    kBypass,
    kLoCutoffFreq,
    kLoCutoffQ,
    kHiCutoffFreq,
    kHiCutoffQ,
    kHDCDBypass,
    kHDCDGainScaleOff,
    kDDXCompBypass,
    kNotch6Freq,
    kNotch6Q,
    kNotch6Cut,
    kNotch7Freq,
    kNotch7Q,
    kNotch7Cut,
    kAnalogIn,
    kAnalogVolume,
    kDelay,
    kAllpassFreq,
    kAllpassQ,
    kBypassMask,
    kHiCutoff2Freq,
    kHiCutoff2Q,
    kNotch8Freq,
    kNotch8Q,
    kNotch8Cut,
    kNotch9Freq,
    kNotch9Q,
    kNotch9Cut,
    kNotchAFreq,
    kNotchAQ,
    kNotchACut,
    kDBNotchWidth -
};

#define kUIArraySize (kDBNotchWidth+1)

// orphaned control IDs
/*
#define kNotch6Freq (10000+20)
#define kNotch6Q (10000+21)
#define kNotch6Boost (10000+22)
#define kNotch7Freq (10000+23)
#define kNotch7Q (10000+24)
#define kNotch7Boost (10000+25)
#define kLoShelf2Freq (10000+26)
#define kLoShelf2Boost (10000+27)
#define kHiShelf2Freq (10000+28)
#define kHiShelf2Boost (10000+29)

```



\*/

// DSP IDs

```
enum {
    kDSPMainVolume,
    kDSPLoShelfFreq,
    kDSPLoShelfGain,
    kDSPHiShelfFreq,
    kDSPHiShelfGain,
    kDSPNotch1Freq,
    kDSPNotch1Q,
    kDSPNotch1Cut,
    kDSPNotch2Freq,
    kDSPNotch2Q,
    kDSPNotch2Cut,
    kDSPNotch3Freq,
    kDSPNotch3Q,
    kDSPNotch3Cut,
    kDSPNotch4Freq,
    kDSPNotch4Q,
    kDSPNotch4Cut,
    kDSPNotch5Freq,
    kDSPNotch5Q,
    kDSPNotch5Cut,
    kDSPPreVolume,
    kDSPBypass,
    kDSPLoCutoffScale,
    kDSPLoCutoffFc,
    kDSPLoCutoffQc,
    kDSPHiCutoffScale,
    kDSPHiCutoffA2,
    kDSPHiCutoffA1,
    kDSPHDCDBypass,
    kDSPHDCDGainScaleOff,
    kSPDDXCompBypass,
    kDSPNotch6Freq,
    kDSPNotch6Q,
    kDSPNotch6Cut,
    kDSPNotch7Freq,
    kDSPNotch7Q,
    kDSPNotch7Cut,
    kDSPAnalogIn,
    kDSPAnalogVol,
    kDSPBypassMask,
    kSPDelay,
    kDSPAllpassFreq,
    kDSPAllpassQ,
    kDSP2HiCutoffScale,
    kDSP2HiCutoffA2,
    kDSP2HiCutoffA1,
    kDSPNotch8Freq,
    kDSPNotch8Q,
    kDSPNotch8Cut,
    kDSPNotch9Freq,
    kDSPNotch9Q,
    kDSPNotch9Cut,
    kDSPNotchAFreq,
    kDSPNotchAQ,
    kDSPNotchACut
};
```

#define kDSPArraySize (kDSPNotchACut+1)

// bit definitions for kDSPBypassMask above

```
enum {
    kBypassNotch1,
    kBypassNotch2,
    kBypassNotch3,
    kBypassNotch4,
    kBypassHipass,
    kBypassLopass,
    kBypassLoshelf,

```

0046

*The Journal of Law, Economics, & Organization*, V16 N1  
 Copyright © 2000 by Oxford University Press  
 All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Oxford University Press.  
 Printed in Great Britain  
 Typeset by Alden Bates Ltd., Exeter, Devon, UK  
 Published by Blackwell Publishers, 108 Cowley Road, Oxford OX4 1JF, UK and 350 Main Street, Malden, MA 02148, USA.

```

#if !defined(AFX_DBNOTCH_H__241CC3C5_14D9_11D3_96EE_006097CDB9E2__INCLUDED_)
#define AFX_DBNOTCH_H__241CC3C5_14D9_11D3_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// DBNotch.h : header file
//

class CTabDialog;

////////////////////////////////////
// CDBNotch dialog

class CDBNotch : public CPropertyPage
{
    DECLARE_DYNCREATE(CDBNotch)

// Construction
public:
    CDBNotch();
    ~CDBNotch();

// Dialog Data
    //{AFX_DATA(CDBNotch)
    enum { IDD = IDD_PP11 };
    CButton m_Bypass;
    CSliderCtrl m_CompGainSlider;
    CSliderCtrl m_CompQSlider;
    CSliderCtrl m_QSlider;
    CSliderCtrl m_FreqSlider;
    //}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{AFX_VIRTUAL(CDBNotch)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
    protected:
    // Generated message map functions
    //{AFX_MSG(CDBNotch)
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnPaint();
    virtual BOOL OnInitDialog();
    afx_msg void OnBypass();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_DBNOTCH_H__241CC3C5_14D9_11D3_96EE_006097CDB9E2__INCLUDED_)

```

```
// DBNotch.cpp : implementation file
//
```

```
#include "stdafx.h"
#include "sa.h"
#include "DBNotch.h"
#include "TabDialog.h"
#include "DSP56kManager.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
////////////////////////////////////
// CDBNotch property page
```

```
IMPLEMENT_DYNCREATE(CDBNotch, CPropertyPage)
```

```
CDBNotch::CDBNotch() : CPropertyPage(CDBNotch::IDD)
{
    //{{AFX_DATA_INIT(CDBNotch)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}
```

```
CDBNotch::~CDBNotch()
{
}
```

```
void CDBNotch::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CDBNotch)
    DDX_Control(pDX, IDC_CHECK5, m_Bypass);
    DDX_Control(pDX, IDC_SLIDER6, m_CompGainSlider);
    DDX_Control(pDX, IDC_SLIDER5, m_CompQSlider);
    DDX_Control(pDX, IDC_SLIDER2, m_QSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_FreqSlider);
    //}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CDBNotch, CPropertyPage)
    //{{AFX_MSG_MAP(CDBNotch)
    ON_WM_VSCROLL()
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_CHECK5, OnBypass)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////
// CDBNotch message handlers
```

```
void CDBNotch::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
```

```
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;
```

```
CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);
```

```
Sleep(50);
if( slider == &m_FreqSlider )
    which = kNotch9Freq;
else if( slider == &m_QSlider )
    which = kNotch9Q;
else if( slider == &m_CompQSlider )
    which = kNotch8Q;
else if( slider == &m_CompGainSlider )
    which = kNotch8Cut;
```

0049

```

else
    return;
g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(), which);
SendStringToUI(which);
}

void CDBNotch::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_FreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch9Freq));
    m_QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch9Q));
    m_CompQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch8Q));
    m_CompGainSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch8Cut));

    m_Bypass.SetCheck(g_DSPManager->GetBypassSection(kBypassDBNotch));

    // Do not call CPropertyPage::OnPaint() for painting messages
}

BOOL CDBNotch::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_FreqSlider.SetRange(0, CONTROL_RANGE);
    m_FreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch9Freq));
    m_FreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_QSlider.SetRange(0, CONTROL_RANGE);
    m_QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch9Q));
    m_QSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_CompQSlider.SetRange(0, CONTROL_RANGE);
    m_CompQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch8Q));
    m_CompQSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_CompGainSlider.SetRange(0, CONTROL_RANGE);
    m_CompGainSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch8Cut));
    m_CompGainSlider.SetTicFreq((CONTROL_RANGE+1)/16);

    m_ParentWindow = (CTabDialog *) GetParent()->GetParent();

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

void CDBNotch::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which, str);
    m_ParentWindow->SetStatusString(0, str);
}

void CDBNotch::OnBypass()
{
    // TODO: Add your control notification handler code here
    int state = m_Bypass.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassDBNotch);
}

```

```

// MainPage.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "MainPage.h"

#include "TabDialog.h"
#include "DSP56kManager.h"
#include <stdlib.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CMainPage property page

IMPLEMENT_DYNCREATE(CMainPage, CPropertyPage)

CMainPage::CMainPage() : CPropertyPage(CMainPage::IDD)
{
    //{{AFX_DATA_INIT(CMainPage)
    m_BypassCheckBox = FALSE;
    m_HDCDBypass = FALSE;
    m_GainScaleBypass = FALSE;
    m_AnalogInput = FALSE;
    //}}AFX_DATA_INIT
}

CMainPage::~CMainPage()
{
}

void CMainPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CMainPage)
    DDX_Control(pDX, IDC_SLIDER8, m_Delay);
    DDX_Control(pDX, IDC_SLIDER7, m_AnalogVol);
    DDX_Control(pDX, IDC_BUTTON1, m_ResetAll);
    DDX_Control(pDX, IDC_SLIDER3, m_PreVolumeSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_VolumeSlider);
    DDX_Check(pDX, IDC_CHECK1, m_BypassCheckBox);
    DDX_Check(pDX, IDC_CHECK2, m_HDCDBypass);
    DDX_Check(pDX, IDC_CHECK3, m_GainScaleBypass);
    DDX_Check(pDX, IDC_CHECK4, m_AnalogInput);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMainPage, CPropertyPage)
    //{{AFX_MSG_MAP(CMainPage)
    ON_WM_VSCROLL()
    ON_WM_SHOWWINDOW()
    ON_BN_CLICKED(IDC_CHECK1, OnBypassButton)
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_CHECK2, OnHDCDBypass)
    ON_BN_CLICKED(IDC_CHECK3, OnGainScaleBypass)
    ON_BN_CLICKED(IDC_BUTTON1, OnResetAll)
    ON_BN_CLICKED(IDC_CHECK4, OnAnalogInput)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CMainPage message handlers

void CMainPage::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)

```

```

{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if ( slider == &m_VolumeSlider)
        which = kMainVolume;
    else if (slider == &m_PreVolumeSlider)
        which = kPreVolume;
    else if (slider == &m_AnalogVol)
        which = kAnalogVolume;
    else if (slider == &m_Delay)
        which = kDelay;
    else
        return;
    g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(), which);
    SendStringToUI(which);
}

BOOL CMainPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_VolumeSlider.SetRange(0, CONTROL_RANGE);
    m_VolumeSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kMainVolume));
    m_VolumeSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_VolumeSlider.SetPageSize(PG_CONTROL_AMT);

    m_PreVolumeSlider.SetRange(0, CONTROL_RANGE);
    m_PreVolumeSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kPreVolume));
    m_PreVolumeSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_PreVolumeSlider.SetPageSize(PG_CONTROL_AMT);

    m_AnalogVol.SetRange(0, CONTROL_RANGE);
    m_AnalogVol.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAnalogVolume));
    m_AnalogVol.SetTicFreq((CONTROL_RANGE+1)/16);
    m_AnalogVol.SetPageSize(PG_CONTROL_AMT);

    m_Delay.SetRange(0, CONTROL_RANGE);
    m_Delay.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAnalogVolume));
    m_Delay.SetTicFreq((CONTROL_RANGE+1)/16);
    m_Delay.SetPageSize(PG_CONTROL_AMT);

    m_BypassCheckBox = false;
    m_ParentWindow = (CDialog *) GetParent()->GetParent();

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

void CMainPage::OnShowWindow(BOOL bShow, UINT nStatus)
{
    CPropertyPage::OnShowWindow(bShow, nStatus);

    // TODO: Add your message handler code here
}

void CMainPage::OnBypassButton()
{
    // TODO: Add your control notification handler code here
    m_BypassCheckBox = !m_BypassCheckBox;
    g_DSPManager->SetBypass(m_BypassCheckBox);
}

void CMainPage::OnPaint()

```

0052

```

{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_VolumeSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kMainVolume));
    m_PreVolumeSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kPreVolume));
    m_AnalogVol.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAnalogVolume));
    m_Delay.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kDelay));

    // Do not call CPropertyPage::OnPaint() for painting messages
}

void CMainPage::OnHDCDBypass()
{
    // TODO: Add your control notification handler code here
    m_HDCDBypass = !m_HDCDBypass;
    g_DSPManager->SetHDCDBypass(m_HDCDBypass);
}

void CMainPage::OnGainScaleBypass()
{
    // TODO: Add your control notification handler code here
    m_GainScaleBypass = !m_GainScaleBypass;
    g_DSPManager->SetHDCDGainScale(m_GainScaleBypass);
}

void CMainPage::OnResetAll()
{
    // TODO: Add your control notification handler code here

    // MM 8/3/99 Added extra dialog warning.
    if( MessageBox("Are you sure you want to reset all parameters?", "WARNING", MB_YESNO) != IDYES )
        return;
    g_DSPManager->ResetAll();
    m_VolumeSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kMainVolume));
    m_PreVolumeSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kPreVolume));
    m_AnalogVol.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kAnalogVolume));
    m_Delay.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kDelay));
}

void CMainPage::OnAnalogInput()
{
    // TODO: Add your control notification handler code here
    m_AnalogInput = !m_AnalogInput;
    g_DSPManager->SetAnalogInput(m_AnalogInput);
}

void CMainPage::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which, str);
    m_ParentWindow->SetStatusString(0, str);
}

```



```

#if !defined(AFX_MAINPAGE_H__90463DA4_D52E_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_MAINPAGE_H__90463DA4_D52E_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif

// _MSC_VER >= 1000
// MainPage.h : header file
//

class CTabDialog;

////////////////////////////////////

// CMainPage dialog

class CMainPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CMainPage)

// Construction
public:
    CMainPage();
    ~CMainPage();

// Dialog Data
    //{AFX_DATA(CMainPage)
    enum { IDD = IDD_PP2 };
    CSliderCtrl m_Delay;
    CSliderCtrl m_AnalogVol;
    CButton m_ResetAll;
    CSliderCtrl m_PreVolumeSlider;
    CSliderCtrl m_VolumeSlider;
    BOOL m_BypassCheckBox;
    BOOL m_HDCDBypass;
    BOOL m_GainScaleBypass;
    BOOL m_AnalogInput;
    //}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{AFX_VIRTUAL(CMainPage)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{AFX_MSG(CMainPage)
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    virtual BOOL OnInitDialog();
    afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
    afx_msg void OnBypassButton();
    afx_msg void OnPaint();
    afx_msg void OnHDCDBypass();
    afx_msg void OnGainScaleBypass();
    afx_msg void OnResetAll();
    afx_msg void OnAnalogInput();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);

private:
};

//{AFX_INSERT_LOCATION}

```

0054

[illegible]

```

// I2CIPortComm.h: interface for the I2CIPortComm class.
//
////////////////////////////////////

#if !defined(AFX_I2CIPORTCOMM_H__33F9EF07_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_I2CIPORTCOMM_H__33F9EF07_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "DSPComm.h"

enum {
    MI2C_STATE_CLOSED,
    MI2C_STATE_PENDING_OPEN,
    MI2C_STATE_OPEN,
    MI2C_STATE_PENDING_TX,
    MI2C_STATE_PENDING_RX
};

#define MI2_BUFFER_SIZE (3*3000) // should be multiple of 3

class I2CIPortComm : public DSPComm
{
    int m_state;
    int m_error_code;
    long m_TxReq;
    char m_tx_buffer[MI2_BUFFER_SIZE];
    long m_tx_write, m_tx_read;
    long m_detect;
    BOOL m_get_detect;
    long m_retryCount;
    BOOL m_initialTry;
    HWND m_HWnd;

public:
    virtual BOOL IsTransmitting(void);
    virtual long GetDetectState(void);
    virtual void MessageHandler(WPARAM iPortEventCode);
    virtual BOOL CheckState(void);
    I2CIPortComm(HWND p);
    virtual ~I2CIPortComm();
    virtual long SendDSPWord(long);
    virtual long SendDSPMemory(char *, long);
};

#endif // !defined(AFX_I2CIPORTCOMM_H__33F9EF07_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_)

```

```

// I2CIPortComm.cpp: implementation of the I2CIPortComm class.
//
/////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sa.h"
#include "I2CIPortComm.h"
#include "COMPortChooser.h"

#include "i2c200.h"

static struct I2C_PROP i2c;
static I2CIPortComm *g_I2CComm = NULL;

#define I2C_SLAVE_ADDRESS 0xB0
#define I2C_MAX_AMOUNT (5*3) // don't send more than this in one-shot
#define I2C_RETRY_COUNT 0

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

//static int cb_message;

void iPortMsgHandler(WPARAM iPortEventCode)
{
    if( g_I2CComm )
        g_I2CComm->MessageHandler(iPortEventCode);
}

/////////////////////////////////////////////////////////////////
// Construction/Destruction
/////////////////////////////////////////////////////////////////
/*static BOOL wait_for_message(void)
{
    long count;

    while( cb_message == -1 )
    {
        Sleep(100);
        count++;
        if( count > 20 )
            return( false );
    }
    return( true );
}*/

I2CIPortComm::I2CIPortComm(HWND p)
{
    int r;
    CCOMPortChooser dlg;

    dlg.DoModal();

    m_error_code = 0;
    m_state = MI2C_STATE_CLOSED;
    m_TxReq = 0;
    m_tx_write = m_tx_read = 0;
    m_detect = 0;
    m_get_detect = false;
    m_initialTry = false;
    m_HWnd = p;

    i2c.comport = dlg.which_port;

```

```

    i2c.WmMsgNo = 0;
    i2c.pfCBF = iPortMsgHandler;
#ifdef NDEBUG
    i2c.pcLogFileName = NULL;
#else
    i2c.pcLogFileName = "log.txt";
#endif
    i2c.LogFileLevel = 1;
    i2c.LogFileSize = 1000;
    i2c.HostSlaveAddr = 0x6E;
    i2c.BusTimeOut = 1000;
    i2c.MasterBitRate = 2; // 100 kHz
    i2c.MasterRxBufSize = 512;
    i2c.MasterTxBufSize = MI2_BUFFER_SIZE;
    i2c.MasterArbRetry = 0;
    i2c.SlaveRxGCEnable = 1;
    i2c.SlaveRxBufSize = 512;
    i2c.SlaveTxBufSize = 512;

    r = I2COpen(m_HWnd,AfxGetInstanceHandle(),&i2c);
    if( r )
        m_error_code = 0x1000;
    m_state = MI2C_STATE_PENDING_OPEN;
    g_I2CComm = this;
}

I2CIPortComm::~I2CIPortComm()
{
    I2CClose();
    g_I2CComm = NULL;
}

long I2CIPortComm::SendDSPWord(long value)
{
    char buffer[3];

    if( m_state < MI2C_STATE_OPEN )
        return( 0L );

    buffer[0] = (value >> 16) & 0xFF;
    buffer[1] = (value >> 8) & 0xFF;
    buffer[2] = value & 0xFF;
    return( SendDSPMemory(buffer,3) );
}

long I2CIPortComm::SendDSPMemory(char *data, long len)
{
    // make sure the link is valid
    if( m_state < MI2C_STATE_OPEN )
        return( 0L );

    long rval = 0L;
    long b;
    long tocopy,amount,freespace;

    // copy data into the tx buffer
    if( data )
    {
        freespace = m_tx_read - m_tx_write;
        if( freespace <= 0 )
            freespace += MI2_BUFFER_SIZE;
        ASSERT( (len + 3) <= freespace ); // not enough space in the buffer
        for( tocopy = len ; tocopy > 0 ; tocopy -= amount )
        {
            if( m_tx_write >= m_tx_read )
            {

```

0058

```

        amount = MI2_BUFFER_SIZE - m_tx_write;
    }
    else
    {
        amount = m_tx_read - m_tx_write;
    }
    amount = min(amount, tocopy);
    memcpy(m_tx_buffer+m_tx_write, data, amount);
    data += amount;
    m_tx_write += amount;
    m_tx_write %= MI2_BUFFER_SIZE;
}
}
// attempt to send it
if( m_state == MI2C_STATE_OPEN )
{
    amount = m_tx_write - m_tx_read;
    // ASSERT( amount == ((amount / 3) * 3) );
    if( amount < 0 )
    {
        amount = MI2_BUFFER_SIZE - m_tx_read;
    }
    // amount = (amount / 3) * 3; // Send complete DSP Words.
    amount = min(amount, I2C_MAX_AMOUNT); // throttle
    b = I2CMasterTx(I2C_SLAVE_ADDRESS, (unsigned char *) m_tx_buffer+m_tx_read, amount, 1);
    if( b == 0 )
    {
        m_state = MI2C_STATE_PENDING_TX;
        m_TxReq = amount;
        if( !m_initialTry )
        {
            m_retryCount = 0;
            m_initialTry = true;
        }
    }
    else
    {
        if( m_error_code == 0 )
        {
            m_error_code = b;
            ASSERT(b == 0);
        }
    }
}
return( rval );
}

void I2CIPortComm::MessageHandler(WPARAM iPortEventCode)
{
    struct I2C_PROP si2c;
    int r;
    long tx_count;
    long val;
    unsigned char buffer[10];

    switch( m_state )
    {
        case MI2C_STATE_PENDING_OPEN:
            if( iPortEventCode == I2C_OPEN_SUCCESSFUL )
            {
                m_state = MI2C_STATE_OPEN;
                if( m_tx_read != m_tx_write )
                {
                    SendDSPMemory(NULL, 0L);
                }
            }
            else
            {
                m_error_code = iPortEventCode;
            }
            break;

        case MI2C_STATE_PENDING_RX:
            if( iPortEventCode == I2C_MRX_COMPLETE )
            {
                r = I2CGetMasterRxMsg(3, buffer);
                if( r == 3 )

```

0059

```

        {
            val = buffer[0] & 0xFF;
            val << 8;
            val = val | (buffer[1] & 0xFF);
            val << 8;
            val = val | (buffer[2] & 0xFF);
            m_detect = val;
        }
    }
    //else
    // m_error_code = iPortEventCode;
    m_state = MI2C_STATE_OPEN;
    if( m_tx_read != m_tx_write )
        SendDSPMemory(NULL, 0L);
    break;

case MI2C_STATE_PENDING_TX:
    if( iPortEventCode == I2C_MTX_COMPLETE )
    {
        // MM 5/20/99 Don't call status here.
        // r = I2CGetStatus(&si2c);
        // tx_count = si2c.MasterTxByteCount;
        tx_count = m_TxReq;
        m_initialTry = false;
    }
    else
    {
        m_retryCount++;
        if( m_retryCount > I2C_RETRY_COUNT )
        {
            m_initialTry = false;
            if( m_error_code == 0 )
                m_error_code = iPortEventCode;
            //ASSERT(m_retryCount <= I2C_RETRY_COUNT);
            I2CClose();
            r = I2COpen(m_HWnd, AfxGetInstanceHandle(), &si2c);
            ASSERT( r == 0 );
            m_state = MI2C_STATE_PENDING_OPEN;
            return;
        }
        tx_count = 0;
    }
    m_state = MI2C_STATE_OPEN;
    m_tx_read += tx_count;
    m_tx_read %= MI2_BUFFER_SIZE;
    if( m_get_detect )
    {
        m_get_detect = false;
        r = I2CMasterRxExt(I2C_SLAVE_ADDRESS, 3, 1, 1);
        if( r )
        {
            if( m_error_code == 0 )
                m_error_code = r;
        }
        else
            m_state = MI2C_STATE_PENDING_RX;
    }
    else if( m_tx_read != m_tx_write )
        SendDSPMemory(NULL, 0L);
    break;

/*
case MI2C_STATE_PENDING_TX:
    if( iPortEventCode == I2C_MTX_COMPLETE )
    {
        r = I2CGetStatus(&si2c);
        if( m_TxReq == si2c.MasterTxByteCount )
        {
            m_state = MI2C_STATE_OPEN;
            m_tx_read += m_TxReq;
            m_tx_read %= MI2_BUFFER_SIZE;
            if( m_tx_read != m_tx_write )
                SendDSPMemory(NULL, 0L);
        }
    }

```

0060

```

    }
    else
    {
        m_error_code = iPortEventCode;
        m_tx_read = m_tx_write = 0;
        m_state = MI2C_STATE_OPEN;
        if( m_tx_read != m_tx_write )
            SendDSPMemory(NULL, 0L);
    }
    break; */
}

}

BOOL I2CIPortComm::CheckState()
{
    return( m_state == MI2C_STATE_OPEN );
}

long I2CIPortComm::GetDetectState()
{
    long r, rval = 0;

    if( m_error_code )
    {
        rval = -m_error_code;
        m_error_code = 0;
    }
    else
    {
        rval = ( m_detect ) ? 1 : 0;
    }

    /*if( m_state == MI2C_STATE_PENDING_TX )
        m_get_detect = true;
    else if( m_state == MI2C_STATE_OPEN )
    {
        r = I2CMasterRxExt(I2C_SLAVE_ADDRESS, 3, 1, 1);
        if( r )
            m_error_code = r;
        else
            m_state = MI2C_STATE_PENDING_RX;
    }*/
    return( rval );
}

BOOL I2CIPortComm::IsTransmitting()
{
    return( m_tx_write != m_tx_read );
}

```

0061



```

#if !defined(AFX_I2CDIALOG_H_B5D510E6_F7D5_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_I2CDIALOG_H_B5D510E6_F7D5_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// I2CDialog.h : header file
//

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CI2CDialog dialog

class CI2CDialog : public CDialog
{
    UINT          m_timerID;
    int           m_state;

// Construction
public:
    CI2CDialog(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{AFX_DATA(CI2CDialog)
    enum { IDD = IDD_DIALOG2 };
    // NOTE: the ClassWizard will add data members here
    //}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CI2CDialog)
    public:
        virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo);
    protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{AFX_MSG(CI2CDialog)
    virtual BOOL OnInitDialog();
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnClose();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{AFX_INSERT_LOCATION}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_I2CDIALOG_H_B5D510E6_F7D5_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0062

```

// I2CDialog.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "I2CDialog.h"
#include "DSP56kManager.h"

#define TIMERID 55

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// CI2CDialog dialog

CI2CDialog::CI2CDialog(CWnd* pParent /*=NULL*/)
: CDialog(CI2CDialog::IDD, pParent)
{
   //{{AFX_DATA_INIT(CI2CDialog)
    // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
}

void CI2CDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CI2CDialog)
    // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CI2CDialog, CDialog)
    {{{AFX_MSG_MAP(CI2CDialog)
    ON_WM_TIMER()
    ON_WM_CLOSE()
    }}}AFX_MSG_MAP
END_MESSAGE_MAP()

//////////////////////////////////////
// CI2CDialog message handlers

BOOL CI2CDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: Add extra initialization here
    g_DSPManager = new CDSP56kManager(this->m_hWnd);

    m_timerID = 0;
    m_timerID = SetTimer(TIMERID, 50, NULL);
    m_state = 0;

    return TRUE; // return TRUE unless you set the focus to a control
    // EXCEPTION: OCX Property Pages should return FALSE
}

void CI2CDialog::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    if( nIDEvent == TIMERID )
    {
        if( g_DSPManager->IsReady() )
        {
            switch( m_state )

```

```

    {
        case 0:
            g_DSPManager->DownloadDSPCode();
            m_state++;
            break;
        case 1:
            g_DSPManager->ResetAll();
            EndDialog(IDOK);
            break;
        default:
            EndDialog(IDOK);
            break;
    }
}
else if( g_DSPManager->GetHDCDMode() < 0 )
    EndDialog(IDABORT);
}
else
{
    CDialog::OnTimer(nIDEvent);
}
}

```

```

BOOL CI2CDialog::OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo)
{
    // TODO: Add your specialized code here and/or call the base class

    return CDialog::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}

void CI2CDialog::OnClose()
{
    // TODO: Add your message handler code here and/or call default
    if( m_timerID )
        KillTimer( m_timerID );

    CDialog::OnClose();
}

```

```

#if !defined(AFX_I2COMMERRORDIALOG_H__76C389E8_F889_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_I2COMMERRORDIALOG_H__76C389E8_F889_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// I2CommErrorDialog.h : header file
//

/////////////////////////////////////////////////////////////////
// CI2CommErrorDialog dialog

class CI2CommErrorDialog : public CDialog
{
// Construction
public:
    CI2CommErrorDialog(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
   //{{AFX_DATA(CI2CommErrorDialog)
    enum { IDD = IDD_DIALOG4 };
    CString m_ErrorCode;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CI2CommErrorDialog)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
   //{{AFX_MSG(CI2CommErrorDialog)
    afx_msg void OnButton1();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_I2COMMERRORDIALOG_H__76C389E8_F889_11D2_96EE_006097CDB9E2__INCLUDED_)

```

```

// I2CCommErrorDialog.cpp implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "I2CCommErrorDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// CI2CCommErrorDialog dialog

CI2CCommErrorDialog::CI2CCommErrorDialog(CWnd* pParent /*=NULL*/)
: CDialog(CI2CCommErrorDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CI2CCommErrorDialog)
    m_ErrorCode = _T("");
    //}}AFX_DATA_INIT
}

void CI2CCommErrorDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CI2CCommErrorDialog)
    DDX_Text(pDX, IDC_EDIT1, m_ErrorCode);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CI2CCommErrorDialog, CDialog)
    //{{AFX_MSG_MAP(CI2CCommErrorDialog)
    ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

//////////////////////////////////////
// CI2CCommErrorDialog message handlers

void CI2CCommErrorDialog::OnButton1()
{
    // TODO: Add your control notification handler code here
    EndDialog(IDCANCEL);
}

```

```

// PEQParam.h: interface for the CPEQParam class.
//
/////////////////////////////////////////////////////////////////

#if !defined(AFX_PEQPARAM_H__733FB7AB_4A49_11D3_96EE_006097CDB9E2__INCLUDED_)
#define AFX_PEQPARAM_H__733FB7AB_4A49_11D3_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

typedef struct _biquadcoef
{
    float c[6];    // b0,b1,b2,a0,a1,a2
} biquadcoef;

class CPEQParam
{
    float m_Gain;    // in dB (+ or -)
    float m_Q;    // in Q units
    float m_Frequency;    // in Hz
    float m_SampleRate;    // in Hz
    double m_pi;

public:
    void GetCoef(biquadcoef *);
    void GetAllpassCoef(biquadcoef *);
    void SetQ(float v) { m_Q = v; }
    void SetQ(CString &str);
    void SetGain(float v) { m_Gain = v; }
    void SetGain(CString &str);
    void SetFreq(float v) { m_Frequency = v; }
    void SetFreq(CString &str);

    CPEQParam();
    virtual ~CPEQParam();
};

#endif // !defined(AFX_PEQPARAM_H__733FB7AB_4A49_11D3_96EE_006097CDB9E2__INCLUDED_)

```

```

// PEQParam.cpp: implementation of the CPEQParam class.
//
/////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sa.h"
#include "PEQParam.h"
#include <stdio.h>
#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

/////////////////////////////////////////////////////////////////
// Construction/Destruction
/////////////////////////////////////////////////////////////////

CPEQParam::CPEQParam()
: m_Frequency(0.5), m_Gain(0.0), m_Q(1.0), m_SampleRate(44100.)
{
    m_pi = acos(-1.);
}

CPEQParam::~CPEQParam()
{
}

void CPEQParam::SetGain(CString & str)
{
    float v;

    sscanf(str, "%f", &v);
    m_Gain = v;
}

void CPEQParam::SetFreq(CString & str)
{
    float v;

    sscanf(str, "%f", &v);
    m_Frequency = v;
}

void CPEQParam::SetQ(CString & str)
{
    float v;

    sscanf(str, "%f", &v);
    m_Q = v;
}

void CPEQParam::GetAllpassCoef(biquadcoef *f)
{
    double gamma, beta, wc, tbeta;
    double normFreq = m_Frequency/m_SampleRate;

    wc = m_pi*normFreq;
    gamma = -cos(wc);
    tbeta = wc/(2*m_Q);
    if( tbeta > m_pi/4 )
        tbeta = m_pi/4;
    beta = (1.-tan(tbeta))/(1.+tan(tbeta));

    f->c[0] = (float) beta;          // b0

```

```

f->c[1] = (float) (gain*(1+beta)); // b1
f->c[2] = 1.; // b2
f->c[3] = 1.; // a0
f->c[4] = f->c[1]; // a1
f->c[5] = (float) beta; // a2
}

void CPEQParam::GetCoef(biquadcoef *f)
{
    double M,L;
    biquadcoef af;
    double gain = (float) pow(10.,m_Gain/20.);

    GetAllpassCoef(&af);
    M = (1.-gain)/2.;
    L = (1.+gain)/2.;

    f->c[0] = (float) (af.c[0]*M+L); // b0
    f->c[1] = (float) (af.c[1]*M+af.c[4]*L); // b1
    f->c[2] = (float) (af.c[2]*M+af.c[5]*L); // b2
    f->c[3] = af.c[3];
    f->c[4] = af.c[4];
    f->c[5] = af.c[5];
}

```



```

#if !defined(AFX_NOTCHPAGE2_H__5C8994C7_E2A4_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_NOTCHPAGE2_H__5C8994C7_E2A4_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// NotchPage2.h : header file
//

class CTabDialog;

////////////////////////////////////

// CNotchPage2 dialog

class CNotchPage2 : public CPropertyPage
{
    DECLARE_DYNCREATE(CNotchPage2)

// Construction
public:
    CNotchPage2();
    ~CNotchPage2();

// Dialog Data
    //{AFX_DATA(CNotchPage2)
    enum { IDD = IDD_PP9 };
    CButton m_BypassSecondButton;
    CButton m_BypassFirstButton;
    CSliderCtrl m_SliderCut4;
    CSliderCtrl m_SliderQ4;
    CSliderCtrl m_SliderFrequency4;
    CSliderCtrl m_SliderCut3;
    CSliderCtrl m_SliderQ3;
    CSliderCtrl m_SliderFrequency3;
    //}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{AFX_VIRTUAL(CNotchPage2)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{AFX_MSG(CNotchPage2)
    afx_msg void OnPaint();
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnBypassFirst();
    afx_msg void OnBypassSecond();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);

};

//{AFX_INSERT_LOCATION}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_NOTCHPAGE2_H__5C8994C7_E2A4_11D2_96EE_006097CDB9E2__INCLUDED_)

```

```

// NotchPage2.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "NotchPage2.h"
#include "TabDialog.h"
#include "DSP56kManager.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CNotchPage2 property page

IMPLEMENT_DYNCREATE(CNotchPage2, CPropertyPage)

CNotchPage2::CNotchPage2() : CPropertyPage(CNotchPage2::IDD)
{
    ///{{AFX_DATA_INIT(CNotchPage2)
    ///}}AFX_DATA_INIT
}

CNotchPage2::~CNotchPage2()
{
}

void CNotchPage2::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    ///{{AFX_DATA_MAP(CNotchPage2)
    DDX_Control(pDX, IDC_CHECK6, m_BypassSecondButton);
    DDX_Control(pDX, IDC_CHECK5, m_BypassFirstButton);
    DDX_Control(pDX, IDC_SLIDER6, m_SliderCut4);
    DDX_Control(pDX, IDC_SLIDER5, m_SliderQ4);
    DDX_Control(pDX, IDC_SLIDER4, m_SliderFrequency4);
    DDX_Control(pDX, IDC_SLIDER3, m_SliderCut3);
    DDX_Control(pDX, IDC_SLIDER2, m_SliderQ3);
    DDX_Control(pDX, IDC_SLIDER1, m_SliderFrequency3);
    ///}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CNotchPage2, CPropertyPage)
    ///{{AFX_MSG_MAP(CNotchPage2)
    ON_WM_PAINT()
    ON_WM_VSCROLL()
    ON_BN_CLICKED(IDC_CHECK5, OnBypassFirst)
    ON_BN_CLICKED(IDC_CHECK6, OnBypassSecond)
    ///}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CNotchPage2 message handlers

void CNotchPage2::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_SliderFrequency3.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Freq));
    m_SliderQ3.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Q));
    m_SliderCut3.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Cut));
    m_SliderFrequency4.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Freq));
    m_SliderQ4.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Q));
    m_SliderCut4.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Cut));

    m_BypassFirstButton.SetCheck(g_DSPManager->GetBypassSection(kBypassNotch3));
    m_BypassSecondButton.SetCheck(g_DSPManager->GetBypassSection(kBypassNotch4));
}

```

```

    // Do not call CPropertyPage::OnPaint() for painting messages
}

BOOL CNotchPage2::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_SliderFrequency3.SetRange(0, CONTROL_RANGE);
    m_SliderFrequency3.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Freq));
    m_SliderFrequency3.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderFrequency3.SetPageSize(PG_CONTROL_AMT);

    m_SliderQ3.SetRange(0, CONTROL_RANGE);
    m_SliderQ3.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Q));
    m_SliderQ3.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderQ3.SetPageSize(PG_CONTROL_AMT);

    m_SliderCut3.SetRange(0, CONTROL_RANGE);
    m_SliderCut3.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch6Cut));
    m_SliderCut3.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderCut3.SetPageSize(PG_CONTROL_AMT);

    m_SliderFrequency4.SetRange(0, CONTROL_RANGE);
    m_SliderFrequency4.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Freq));
    m_SliderFrequency4.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderFrequency4.SetPageSize(PG_CONTROL_AMT);

    m_SliderQ4.SetRange(0, CONTROL_RANGE);
    m_SliderQ4.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Q));
    m_SliderQ4.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderQ4.SetPageSize(PG_CONTROL_AMT);

    m_SliderCut4.SetRange(0, CONTROL_RANGE);
    m_SliderCut4.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch7Cut));
    m_SliderCut4.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderCut4.SetPageSize(PG_CONTROL_AMT);

    m_ParentWindow = (CTabDialog *) GetParent()->GetParent();

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

void CNotchPage2::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if( slider == &m_SliderFrequency3 )
        which = kNotch6Freq;
    else if( slider == &m_SliderQ3 )
        which = kNotch6Q;
    else if( slider == &m_SliderCut3 )
        which = kNotch6Cut;
    else if( slider == &m_SliderFrequency4 )
        which = kNotch7Freq;
    else if( slider == &m_SliderQ4 )
        which = kNotch7Q;
    else if( slider == &m_SliderCut4 )
        which = kNotch7Cut;
    else
        return;

    g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(), which);
    SendStringToUI(which);
}

```

0072

```

void CNotchPage2::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which, str);
    m_ParentWindow->SetStatusString(0, str);
}

void CNotchPage2::OnBypassFirst()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassFirstButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassNotch3);
}

void CNotchPage2::OnBypassSecond()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassSecondButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassNotch4);
}

```

0073

```

#if !defined(AFX_NOTCHPAGE1_H__OCF7E208_D790_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_NOTCHPAGE1_H__OCF7E208_D790_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// NotchPage1.h : header file
//

////////////////////////////////////
// CNotchPage1 dialog

class CTabDialog;

class CNotchPage1 : public CPropertyPage
{
    DECLARE_DYNCREATE(CNotchPage1)

// Construction
public:
    CNotchPage1();
    ~CNotchPage1();

// Dialog Data
    //{AFX_DATA(CNotchPage1)
    enum { IDD = IDD_PP4 };
    CButton m_BypassSecondButton;
    CButton m_BypassFirstButton;
    CSliderCtrl m_SliderCut2;
    CSliderCtrl m_SliderQ2;
    CSliderCtrl m_SliderFrequency2;
    CSliderCtrl m_SliderCut1;
    CSliderCtrl m_SliderQ1;
    CSliderCtrl m_SliderFrequency1;
    //}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{AFX_VIRTUAL(CNotchPage1)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{AFX_MSG(CNotchPage1)
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
    afx_msg void OnPaint();
    afx_msg void OnBypassFirst();
    afx_msg void OnBypassSecond();
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);

};

//{AFX_INSERT_LOCATION}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_NOTCHPAGE1_H__OCF7E208_D790_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0074

```
// NotchPage1.cpp : implementation file
//
```

```
#include "stdafx.h"
#include "sa.h"
#include "NotchPage1.h"
#include "TabDialog.h"
#include "DSP56kManager.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
////////////////////////////////////
// CNotchPage1 property page
```

```
IMPLEMENT_DYNCREATE(CNotchPage1, CPropertyPage)
```

```
CNotchPage1::CNotchPage1() : CPropertyPage(CNotchPage1::IDD)
{
    //{{AFX_DATA_INIT(CNotchPage1)
    //}}AFX_DATA_INIT
}
```

```
CNotchPage1::~CNotchPage1()
{
}
```

```
void CNotchPage1::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CNotchPage1)
    DDX_Control(pDX, IDC_CHECK6, m_BypassSecondButton);
    DDX_Control(pDX, IDC_CHECK5, m_BypassFirstButton);
    DDX_Control(pDX, IDC_SLIDER6, m_SliderCut2);
    DDX_Control(pDX, IDC_SLIDER5, m_SliderQ2);
    DDX_Control(pDX, IDC_SLIDER4, m_SliderFrequency2);
    DDX_Control(pDX, IDC_SLIDER3, m_SliderCut1);
    DDX_Control(pDX, IDC_SLIDER2, m_SliderQ1);
    DDX_Control(pDX, IDC_SLIDER1, m_SliderFrequency1);
    //}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CNotchPage1, CPropertyPage)
    //{{AFX_MSG_MAP(CNotchPage1)
    ON_WM_VSCROLL()
    ON_WM_SHOWWINDOW()
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_CHECK5, OnBypassFirst)
    ON_BN_CLICKED(IDC_CHECK6, OnBypassSecond)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////
// CNotchPage1 message handlers
```

```
BOOL CNotchPage1::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_SliderFrequency1.SetRange(0, CONTROL_RANGE);
    m_SliderFrequency1.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch1Freq));
    m_SliderFrequency1.SetTicFreq((CONTROL_RANGE+1)/16);
    m_SliderFrequency1.SetPageSize(PG_CONTROL_AMT);

    m_SliderQ1.SetRange(0, CONTROL_RANGE);
    m_SliderQ1.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch1Q));
    m_SliderQ1.SetTicFreq((CONTROL_RANGE+1)/16);
```

0075

```

m_SliderQ1.SetPageSize(PG_CONTROL_AMT);

m_SliderCut1.SetRange(0,CONTROL_RANGE);
m_SliderCut1.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch1Cut));
m_SliderCut1.SetTicFreq((CONTROL_RANGE+1)/16);
m_SliderCut1.SetPageSize(PG_CONTROL_AMT);

m_SliderFrequency2.SetRange(0,CONTROL_RANGE);
m_SliderFrequency2.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch2Freq));
m_SliderFrequency2.SetTicFreq((CONTROL_RANGE+1)/16);
m_SliderFrequency2.SetPageSize(PG_CONTROL_AMT);

m_SliderQ2.SetRange(0,CONTROL_RANGE);
m_SliderQ2.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch2Q));
m_SliderQ2.SetTicFreq((CONTROL_RANGE+1)/16);
m_SliderQ2.SetPageSize(PG_CONTROL_AMT);

m_SliderCut2.SetRange(0,CONTROL_RANGE);
m_SliderCut2.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch2Cut));
m_SliderCut2.SetTicFreq((CONTROL_RANGE+1)/16);
m_SliderCut2.SetPageSize(PG_CONTROL_AMT);

m_ParentWindow = (CDialog *) GetParent()->GetParent();

return TRUE; // return TRUE unless you set the focus to a control
// EXCEPTION: OCX Property Pages should return FALSE
}

void CNotchPage1::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if( slider == &m_SliderFrequency1 )
        which = kNotch1Freq;
    else if( slider == &m_SliderQ1 )
        which = kNotch1Q;
    else if( slider == &m_SliderCut1 )
        which = kNotch1Cut;
    else if( slider == &m_SliderFrequency2 )
        which = kNotch2Freq;
    else if( slider == &m_SliderQ2 )
        which = kNotch2Q;
    else if( slider == &m_SliderCut2 )
        which = kNotch2Cut;
    else
        return;
    g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(),which);
    SendStringToUI(which);
}

void CNotchPage1::OnShowWindow(BOOL bShow, UINT nStatus)
{
    CPropertyPage::OnShowWindow(bShow, nStatus);

    // TODO: Add your message handler code here
}

void CNotchPage1::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_SliderFrequency1.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch1Freq));
    m_SliderQ1.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch1Q));
    m_SliderCut1.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch1Cut));

```

0076

```

m_SliderFrequency2.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch2Freq));
m_SliderQ2.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch2Q));
m_SliderCut2.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch2Cut));
m_BypassFirstButton.SetCheck(g_DSPManager->GetBypassSection(kBypassNotch1));
m_BypassSecondButton.SetCheck(g_DSPManager->GetBypassSection(kBypassNotch2));

// Do not call CPropertyPage::OnPaint() for painting messages
}

void CNotchPage1::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which,str);
    m_ParentWindow->SetStatusString(0,str);
}

void CNotchPage1::OnBypassFirst()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassFirstButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassNotch1);
}

void CNotchPage1::OnBypassSecond()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassSecondButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassNotch2);
}

```

0077



```

#if !defined(AFX_NEWCUTOFFPAGE_H__E2728226_E026_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_NEWCUTOFFPAGE_H__E2728226_E026_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// NewCutoffPage.h : header file
//

class CTabDialog;

////////////////////////////////////
// CNewCutoffPage dialog

class CNewCutoffPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CNewCutoffPage)

// Construction
public:
    CNewCutoffPage();
    ~CNewCutoffPage();

// Dialog Data
    //{{AFX_DATA(CNewCutoffPage)
    enum { IDD = IDD_PP8 };
    CButton m_BypassNlopassButton;
    CSliderCtrl m_NewHiQSlider;
    CSliderCtrl m_NewHiFreqSlider;
    CButton m_BypassLopassButton;
    CButton m_BypassHipassButton;
    CSliderCtrl m_HiQSlider;
    CSliderCtrl m_HiFreqSlider;
    CSliderCtrl m_LoQSlider;
    CSliderCtrl m_LoFreqSlider;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CNewCutoffPage)
    public:
        virtual BOOL OnSetActive();
    protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CNewCutoffPage)
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    virtual BOOL OnInitDialog();
    afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
    afx_msg void OnPaint();
    afx_msg void OnBypassHipass();
    afx_msg void OnBypassLopass();
    afx_msg void OnBypassNewLopass();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_NEWCUTOFFPAGE_H__E2728226_E026_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0078

```

// NewCutoffPage.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "NewCutoffPage.h"
#include "TabDialog.h"
#include "DSP56kManager.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CNewCutoffPage property page

IMPLEMENT_DYNCREATE(CNewCutoffPage, CPropertyPage)

CNewCutoffPage::CNewCutoffPage() : CPropertyPage(CNewCutoffPage::IDD)
{
    ///{{AFX_DATA_INIT(CNewCutoffPage)
    ///}}AFX_DATA_INIT
}

CNewCutoffPage::~CNewCutoffPage()
{
}

void CNewCutoffPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    ///{{AFX_DATA_MAP(CNewCutoffPage)
    DDX_Control(pDX, IDC_CHECK7, m_BypassNlopassButton);
    DDX_Control(pDX, IDC_SLIDER9, m_NewHiQSlider);
    DDX_Control(pDX, IDC_SLIDER8, m_NewHiFreqSlider);
    DDX_Control(pDX, IDC_CHECK5, m_BypassLopassButton);
    DDX_Control(pDX, IDC_CHECK4, m_BypassHipassButton);
    DDX_Control(pDX, IDC_SLIDER5, m_HiQSlider);
    DDX_Control(pDX, IDC_SLIDER4, m_HiFreqSlider);
    DDX_Control(pDX, IDC_SLIDER2, m_LoQSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_LoFreqSlider);
    ///}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CNewCutoffPage, CPropertyPage)
    ///{{AFX_MSG_MAP(CNewCutoffPage)
    ON_WM_VSCROLL()
    ON_WM_SHOWWINDOW()
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_CHECK4, OnBypassHipass)
    ON_BN_CLICKED(IDC_CHECK5, OnBypassLopass)
    ON_BN_CLICKED(IDC_CHECK7, OnBypassNewLopass)
    ///}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CNewCutoffPage message handlers

void CNewCutoffPage::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if ( slider == &m_LoFreqSlider )
        which = kLoCutoffFreq;

```

```

else if( slider == &m_LoQSlider )
    which = kLoCutoffQ;
else if( slider == &m_HiFreqSlider )
    which = kHiCutoffFreq;
else if( slider == &m_HiQSlider )
    which = kHiCutoffQ;
else if( slider == &m_NewHiFreqSlider )
    which = kHiCutoff2Freq;
else if( slider == &m_NewHiQSlider )
    which = kHiCutoff2Q;
else
    return;
g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(), which);
SendStringToUI(which);
}

void CNewCutoffPage::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which, str);
    m_ParentWindow->SetStatusString(0, str);
}

BOOL CNewCutoffPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_LoFreqSlider.SetRange(0, CONTROL_RANGE);
    m_LoFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoCutoffFreq));
    m_LoFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_LoFreqSlider.SetPageSize(PG_CONTROL_AMT);

    m_LoQSlider.SetRange(0, CONTROL_RANGE);
    m_LoQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoCutoffQ));
    m_LoQSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_LoQSlider.SetPageSize(PG_CONTROL_AMT);

    m_HiFreqSlider.SetRange(0, CONTROL_RANGE);
    m_HiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoffFreq));
    m_HiFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_HiFreqSlider.SetPageSize(PG_CONTROL_AMT);

    m_HiQSlider.SetRange(0, CONTROL_RANGE);
    m_HiQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoffQ));
    m_HiQSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_HiQSlider.SetPageSize(PG_CONTROL_AMT);

    m_NewHiFreqSlider.SetRange(0, CONTROL_RANGE);
    m_NewHiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoff2Freq));
    m_NewHiFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_NewHiFreqSlider.SetPageSize(PG_CONTROL_AMT);

    m_NewHiQSlider.SetRange(0, CONTROL_RANGE);
    m_NewHiQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoff2Q));
    m_NewHiQSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_NewHiQSlider.SetPageSize(PG_CONTROL_AMT);

    m_ParentWindow = (CTabDialog *) GetParent()->GetParent();

    return TRUE; // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return FALSE
}

void CNewCutoffPage::OnShowWindow(BOOL bShow, UINT nStatus)
{
    CPropertyPage::OnShowWindow(bShow, nStatus);
}

```

0080

```

// TODO: Add your message handler code here
}

BOOL CNewCutoffPage::OnSetActive()
{
    // TODO: Add your specialized code here and/or call the base class

    return CPropertyPage::OnSetActive();
}

void CNewCutoffPage::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_LoFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoCutoffFreq));
    m_LoQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoCutoffQ));
    m_HiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoffFreq));
    m_HiQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoffQ));
    m_NewHiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoff2Freq));
    m_NewHiQSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiCutoff2Q));

    m_BypassHipassButton.SetCheck(g_DSPManager->GetBypassSection(kBypassHipass));
    m_BypassLopassButton.SetCheck(g_DSPManager->GetBypassSection(kBypassLopass));
    m_BypassNlopassButton.SetCheck(g_DSPManager->GetBypassSection(kBypassNlopass));
    // Do not call CPropertyPage::OnPaint() for painting messages
}

void CNewCutoffPage::OnBypassHipass()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassHipassButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassHipass);
}

void CNewCutoffPage::OnBypassLopass()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassLopassButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassLopass);
}

void CNewCutoffPage::OnBypassNewLopass()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassNlopassButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state,kBypassNlopass);
}

```

```

// ShelfEQParam.h: interface for the CSelfEQParam class.
//
////////////////////////////////////

#ifdef AFX_SHELFEQPARAM_H__733FB7AB_4A49_11D3_96EE_006097CDB9E2__INCLUDED_
#define AFX_SHELFEQPARAM_H__733FB7AB_4A49_11D3_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

typedef struct _biquadcoef
{
    float c[6]; // b0,b1,b2,a0,a1,a2
} biquadcoef;

class CSelfEQParam
{
    float m_Gain; // in dB (+ or -)
    float m_Frequency; // in Hz
    float m_SampleRate; // in Hz
    double m_pi;
    bool m_HiShelf;

public:
    void GetCoef(biquadcoef *);
    void GetAllpassCoef(biquadcoef *);
    void SetGain(float v) { m_Gain = v; }
    void SetGain(CString &str);
    void SetFreq(float v) { m_Frequency = v; }
    void SetFreq(CString &str);

    CSelfEQParam(bool hi);
    virtual ~CSelfEQParam();
};

#endif // !defined(AFX_SHELFEQPARAM_H__733FB7AB_4A49_11D3_96EE_006097CDB9E2__INCLUDED_)

```

```

// ShelfEQParam.cpp: implementation of the CShelfEQParam class.
//
////////////////////////////////////
#include "stdafx.h"
#include "sa.h"
#include "ShelfEQParam.h"
#include <stdio.h>
#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

CShelfEQParam::CShelfEQParam(bool hi)
: m_Frequency(0.5), m_Gain(0.0), m_SampleRate(44100.), m_HiShelf(hi)
{
    m_pi = acos(-1.);
}

CShelfEQParam::~CShelfEQParam()
{
}

void CShelfEQParam::SetGain(CString & str)
{
    float v;

    sscanf(str, "%f", &v);
    m_Gain = v;
}

void CShelfEQParam::SetFreq(CString & str)
{
    float v;

    sscanf(str, "%f", &v);
    m_Frequency = v;
}

void CShelfEQParam::GetAllpassCoef(biquadcoef *f)
{
    double gamma, wc;
    double normFreq = m_Frequency/m_SampleRate/2;

    wc = m_pi*normFreq;
    gamma = (tan(wc/2)-1)/(tan(wc/2)+1);

    if( m_HiShelf )
    {
        f->c[0] = (float) gamma;           // b0
        f->c[1] = 1.0;                    // b1
    }
    else
    {
        f->c[0] = (float) -gamma;          // b0
        f->c[1] = -1.0;                   // b1
    }

    f->c[2] = 0;                          // b2
    f->c[3] = 1.;                          // a0
    f->c[4] = (float) gamma;              // a1
}

```

0083

```

f->c[5] = 0; // a2
}

void CShelfEQParam::GetCoef(biquadcoef *f)
{
    double M,L;
    biquadcoef af;
    double gain = (float) pow(10.,m_Gain/20.);

    GetAllpassCoef(&af);
    M = (1.-gain)/2.;
    L = (1.+gain)/2.;

    f->c[0] = (float) (af.c[0]*M+L); // b0
    f->c[1] = (float) (af.c[1]*M+af.c[4]*L); // b1
    f->c[2] = (float) (af.c[2]*M+af.c[5]*L); // b2
    f->c[3] = af.c[3];
    f->c[4] = af.c[4];
    f->c[5] = af.c[5];
}

```

```

// saDlg.h : header file
//

#if !defined(AFX_SADLG_H__33D93B0B_D0B8_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_SADLG_H__33D93B0B_D0B8_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

/////////////////////////////////////////////////////////////////
// CSaDlg dialog

class CSaDlg : public CDialog
{
// Construction
public:
    CSaDlg(CWnd* pParent = NULL); // standard constructor

// Dialog Data
//{{AFX_DATA(CSaDlg)
enum { IDD = IDD_SA_DIALOG };
    // NOTE: the ClassWizard will add data members here
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CSaDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
    HICON m_hIcon;

    // Generated message map functions
    {{{AFX_MSG(CSaDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnButton1();
    afx_msg void OnButton2();
    }}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_SADLG_H__33D93B0B_D0B8_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0085



```

// saDlg.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "saDlg.h"

#include "unit_ppi.h"
#include "functs.h"
#include "TabDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

static int cmd_delay = 1000;
static int io_delay = 50;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

    // Dialog Data
    //{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CAboutDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

    // Implementation
protected:
    //{AFX_MSG(CAboutDlg)
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{AFX_DATA_INIT(CAboutDlg)
    //}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{AFX_DATA_MAP(CAboutDlg)
    //}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{AFX_MSG_MAP(CAboutDlg)
    // No message handlers
    //}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CSaDlg dialog

CSaDlg::CSaDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CSaDlg::IDD, pParent)

```

0086

```

{
    //{AFX_DATA_INIT(CSAdlg)
    // NOTE: the ClassWizard will add member initialization here
    //{AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CSAdlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{AFX_DATA_MAP(CSAdlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    //{AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSAdlg, CDialog)
    //{AFX_MSG_MAP(CSAdlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
    ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
    //{AFX_MSG_MAP
END_MESSAGE_MAP()

//////////////////////
// CSAdlg message handlers

BOOL CSAdlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog. The framework does this automatically
    // when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE); // Set big icon
    SetIcon(m_hIcon, FALSE); // Set small icon

    // TODO: Add extra initialization here
    if( load_dll()==false )
    {
        MessageBox("UNIT_PPI not found in the system directory. Fix this and restart application.");
        return false;
    }
    init_port_spi(1);
    set_cmd_delay_cnt_value(cmd_delay);
    set_io_delay_cnt_value(io_delay);

    return TRUE; // return TRUE unless you set the focus to a control
}

void CSAdlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)

```

0087

```

    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

// If you add a minimize button to your dialog, you will need the code below
// to draw the icon. For MFC applications using the document/view model,
// this is automatically done for you by the framework.

void CSaDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CSaDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CSaDlg::OnButton1()
{
    long tlong;

    tlong=spi_xchg24(0xAA55AA);
    tlong=spi_xchg24(0x0);
}

void CSaDlg::OnButton2()
{
    // long tlong;
    //
    // tlong=spi_xchg24(0x654321);
    // tlong=spi_xchg24(0x0);

    CTabDialog dlg;

    dlg.DoModal();
}

```

0088

```

// sa.h : main header file for the SA application
//

#ifndef AFX_SA_H__33D93B09_D0B8_11D2_96EE_006097CDB9E2__INCLUDED_
#define AFX_SA_H__33D93B09_D0B8_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h" // main symbols

////////////////////

// CSAApp:
// See sa.cpp for the implementation of this class
//

class CSAApp : public CWinApp
{
public:
    CSAApp();

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CSAApp)
public:
    virtual BOOL InitInstance();
//}}AFX_VIRTUAL

// Implementation

//{{AFX_MSG(CSAApp)
// NOTE - the ClassWizard will add and remove member functions here.
// DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_SA_H__33D93B09_D0B8_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0089

```

// sa.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "sa.h"
#include "saDlg.h"
#include "TabDialog.h"
#include "I2CDialog.h"
#include "DSP56kManager.h"

CDSP56kManager *g_DSPManager;

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CSaApp

BEGIN_MESSAGE_MAP(CSaApp, CWinApp)
//{{AFX_MSG_MAP(CSaApp)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG
ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

////////////////////////////////////
// CSaApp construction

CSaApp::CSaApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance

    //////////////////////////////////////
    // The one and only CSaApp object

    CSaApp theApp;

    //////////////////////////////////////
    // CSaApp initialization

    BOOL CSaApp::InitInstance()
    {
        AfxEnableControlContainer();

        // Standard initialization
        // If you are not using these features and wish to reduce the size
        // of your final executable, you should remove from the following
        // the specific initialization routines you do not need.

#ifdef _AFXDLL
        Enable3dControls(); // Call this when using MFC in a shared DLL
#else
        Enable3dControlsStatic(); // Call this when linking to MFC statically
#endif

        // Set dialog background color to black
        // SetDialogBkColor(RGB(0,0,0),RGB(255,255,255));

        int response;
        CI2CDialog idlg;
        try
        {
            if( (response = idlg.DoModal()) != IDOK )
            {

```

0090

```

        if( response == IDABORT )
        {
            AfxMessageBox("Trouble initializing");
            Sleep(1000);
        }
//        return FALSE;
    }

    catch(CException* e )
    {
//        printf("Trouble initializing\n");
//        Sleep(1000);
        return FALSE;
    }

// CSaDlg dlg;
CTabDialog dlg;

m_pMainWnd = &dlg;
int nResponse = dlg.DoModal();
if (nResponse == IDOK)
{
    // TODO: Place code here to handle when the dialog is
    // dismissed with OK
}
else if (nResponse == IDCANCEL)
{
    // TODO: Place code here to handle when the dialog is
    // dismissed with Cancel
}

// Since the dialog has been closed, return FALSE so that we exit the
// application, rather than start the application's message pump.
return FALSE;
}

```

0091

```

//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by sa.rc
//
#define IDM_ABOUTBOX                0x0010
#define IDD_ABOUTBOX                100
#define IDS_ABOUTBOX                101
#define IDD_SA_DIALOG               102
#define IDR_MAINFRAME               128
#define IDD_DIALOG1                 129
#define IDD_PP1                     130
#define IDR_MENU1                   130
#define IDD_PP2                     131
#define IDD_PP3                     132
#define IDD_PP4                     133
#define IDD_PP5                     134
#define IDD_PP6                     135
#define IDD_PP7                     136
#define IDD_PP8                     137
#define IDD_PP9                     138
#define IDD_DIALOG2                 139
#define IDD_DIALOG3                 140
#define IDD_DIALOG4                 141
#define IDD_PP10                    142
#define IDD_PP11                    143
#define IDC_BUTTON1                 1001
#define IDC_BUTTON2                 1002
#define IDC_SLIDER1                 1004
#define IDC_CHECK1                  1005
#define IDC_SLIDER2                 1005
#define IDC_SLIDER3                 1006
#define IDC_RADIO1                  1006
#define IDC_SLIDER4                 1007
#define IDC_CHECK2                  1007
#define IDC_RADIO2                  1007
#define IDC_EDIT1                   1007
#define IDC_SLIDER5                 1008
#define IDC_CHECK3                  1008
#define IDC_RADIO3                  1008
#define IDC_SLIDER6                 1009
#define IDC_CHECK4                  1009
#define IDC_RADIO4                  1009
#define IDC_SLIDER7                 1010
#define IDC_CHECK5                  1010
#define IDC_SLIDER8                 1011
#define IDC_CHECK6                  1011
#define IDC_SLIDER9                 1012
#define IDC_CHECK7                  1013
#define ID_MENUITEM32771            32771
#define ID_MENUITEM32772            32772
#define ID_FILE_EXPORTPARAMFILE     32773

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE    142
#define _APS_NEXT_COMMAND_VALUE    32774
#define _APS_NEXT_CONTROL_VALUE    1011
#define _APS_NEXT_SYMED_VALUE      103
#endif
#endif

```

0092

```

#if !defined(AFX_TABDIALOG_H__5CB3DF04_D20F_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_TABDIALOG_H__5CB3DF04_D20F_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// TabDialog.h : header file
//

#include "Page1.h"
#include "MainPage.h"
#include "DDX.h"
#include "NotchPage1.h"
#include "NotchPage2.h"
#include "StWaveRejPage.h"
#include "ShelvPage.h"
#include "NewCutoffPage.h"
#include "AllpassPage.h"
#include "DBNotch.h"

////////////////////////////////////

// CTabDialog dialog

class CTabDialog : public CDialog
{
// Construction
public:
    void DisplayI2CState(long error);
    virtual void SetStatusString(int which, CString &s);
    CTabDialog(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{AFX_DATA(CTabDialog)
    enum { IDD = IDD_DIALOG1 };
    // NOTE: the ClassWizard will add data members here
    //}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CTabDialog)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}AFX_VIRTUAL

// Implementation
protected:
    virtual void DisplayMode(int);

    CStatusBar        m_StatusBar;
    CPropertySheet    m_dlgPropSheet;
    HICON              m_hIcon;
//    CPage1            m_Page1;
    CMainPage         m_MainPage;
    CDDX               m_DDXPage;
    CNotchPage1       m_Notch1Page;
    CNotchPage2       m_Notch2Page;
    CStWaveRejPage    m_StWaveRejPage;
    CShelvPage        m_ShelvPage;
//    CCutoffPage       m_CutoffPage;
    CNewCutoffPage    m_CutoffPage;
    CAllpassPage      m_AllpassPage;
    CDBNotch          m_DBNotch;
    CBrush             m_brush;
    UINT               m_timerID;
    BOOL               m_HDCDDisplay;
    int                m_AIdx, m_ICnt;
    int                m_ErrorCounter;

// Generated message map functions
    //{AFX_MSG(CTabDialog)

```

0093



```

virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
afx_msg void OnClose();
afx_msg void OnMove(int x, int y);
afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor);
afx_msg void OnOpenMenu();
afx_msg void OnSaveMenu();
afx_msg void OnTimer(UINT nIDEvent);
afx_msg void OnFileExportparamfile();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_TABDIALOG_H__5CB3DF04_D20F_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0094

```

// TabDialog.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "TabDialog.h"
#include "DSP56kManager.h"
#include "I2CCommErrorDialog.h"
#include <afxdlgs.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define TIMERID 1
#define ID_HDCD_DETECT 56000
#define ID_ERROR_DISPLAY (ID_HDCD_DETECT+1)

static UINT auIDStatusBar[] = {
    ID_SEPARATOR,
    ID_HDCD_DETECT,
    ID_SEPARATOR,
    ID_ERROR_DISPLAY
};

#define AARRAYSIZE 16
static char *Detect_Array[AARRAYSIZE] = {
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "HDCD",
    "DCD",
    "CD",
    "D",
    "HDCD"
};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// CTabAboutDlg dialog used for App About

class CTabAboutDlg : public CDialog
{
public:
    CTabAboutDlg();

// Dialog Data
//{{AFX_DATA(CTabAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CTabAboutDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
//{{AFX_MSG(CTabAboutDlg)

```

0095

```

    ///}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CTabAboutDlg::CTabAboutDlg() : CDialog(CTabAboutDlg::IDD)
{
    ///{{AFX_DATA_INIT(CTabAboutDlg)
    ///}}AFX_DATA_INIT
}

void CTabAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    ///{{AFX_DATA_MAP(CTabAboutDlg)
    ///}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CTabAboutDlg, CDialog)
    ///{{AFX_MSG_MAP(CTabAboutDlg)
    // No message handlers
    ///}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CTabDialog dialog

CTabDialog::CTabDialog(CWnd* pParent /*=NULL*/)
: CDialog(CTabDialog::IDD, pParent)
{
    ///{{AFX_DATA_INIT(CTabDialog)
    // NOTE: the ClassWizard will add member initialization here
    ///}}AFX_DATA_INIT
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CTabDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    ///{{AFX_DATA_MAP(CTabDialog)
    // NOTE: the ClassWizard will add DDX and DDV calls here
    ///}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CTabDialog, CDialog)
    ///{{AFX_MSG_MAP(CTabDialog)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_CLOSE()
    ON_WM_MOVE()
    ON_WM_CTLCOLOR()
    ON_COMMAND(ID_MENUITEM32771, OnOpenMenu)
    ON_COMMAND(ID_MENUITEM32772, OnSaveMenu)
    ON_WM_TIMER()
    ON_COMMAND(ID_FILE_EXPORTPARAMFILE, OnFileExportparamfile)
    ///}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CTabDialog message handlers

#define PS_WIDTH 222
#define PS_HEIGHT 215

BOOL CTabDialog::OnInitDialog()
{
    int partsList[] = { 50, 100, 150, -1};

```

0096

```

CDialog::OnInitDialog();

// Add "About..." menu item to system menu.

// IDM_ABOUTBOX must be in the system command range.
ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
    CString strAboutMenu;
    strAboutMenu.LoadString(IDS_ABOUTBOX);
    if (!strAboutMenu.IsEmpty())
    {
        pSysMenu->AppendMenu(MF_SEPARATOR);
        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
    }
}

// Set the icon for this dialog. The framework does this automatically
// when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE); // Set big icon
SetIcon(m_hIcon, FALSE); // Set small icon

// build brush
m_brush.CreateSolidBrush(RGB(0,0,0));

// build prop sheet
m_dlgPropSheet.AddPage(&m_MainPage);
m_dlgPropSheet.AddPage(&m_DDXPage);
m_dlgPropSheet.AddPage(&m_ShelvPage);
m_dlgPropSheet.AddPage(&m_CutoffPage);
m_dlgPropSheet.AddPage(&m_Notch1Page);
m_dlgPropSheet.AddPage(&m_Notch2Page);
m_dlgPropSheet.AddPage(&m_StWaveRejPage);
m_dlgPropSheet.AddPage(&m_AllpassPage);
m_dlgPropSheet.AddPage(&m_DBNotch);

m_dlgPropSheet.Create(this, WS_CHILD | WS_VISIBLE, 0);
m_dlgPropSheet.ModifyStyleEx(0, WS_EX_CONTROLPARENT);
m_dlgPropSheet.ModifyStyle(0, WS_TABSTOP);

m_dlgPropSheet.SetWindowPos(NULL, 0, 0, PS_WIDTH, PS_HEIGHT, SWP_NOZORDER | SWP_NOSIZE | SWP_NOACTIVATE);

// resize dialog to fit propsheet exactly
RECT rect;
m_dlgPropSheet.GetWindowRect(&rect);
//rect.bottom += 14;
//rect.right += 14;
SetWindowPos(&wndBottom, rect.left, rect.top, rect.right, rect.bottom, SWP_NOZORDER | SWP_NOMOVE);

// Add status bar
RECT sizerect;
sizerect.top = PS_HEIGHT;
sizerect.bottom = sizerect.top + 20;
sizerect.left = 0;
sizerect.right = PS_WIDTH;
//m_StatusBar.Create(WS_CHILD | WS_VISIBLE | CCS_BOTTOM, sizerect, this, AFX_IDW_STATUS_BAR);
//m_StatusBar.ShowWindow(SW_SHOWNA);
//m_StatusBar.SetSimple();
//m_StatusBar.SetText("Hello world!", 255, 0);
m_StatusBar.Create(this);
//m_StatusBar.SetIndicators((auIDStatusBar, sizeof(auIDStatusBar)/sizeof(UINT));
//m_StatusBar.SetPaneInfo(0, m_StatusBar.GetItemID(0), SBPS_STRETCH, NULL);
//m_StatusBar.GetStatusBarCtrl().SetText("Hello world!", 0, 0);

CRect rcClientStart;
CRect rcClientNow;
GetClientRect(rcClientStart);

```

```
RepositionBars(AFX_IDW_CONTROLBAR_FIRST, AFX_IDW_CONTROLBAR_LAST,
0, reposQuery, rcClientNow);
```

```
// Now move all the controls so they are in the same relative
// position within the remaining client area as they would be
// with no control bars.
/* CPoint ptOffset(rcClientNow.left - rcClientStart.left,
rcClientNow.top - rcClientStart.top);
```

```
CRect rcChild;
CWnd* pwndChild = GetWindow(GW_CHILD);
while (pwndChild)
{
    pwndChild->GetWindowRect(rcChild);
    ScreenToClient(rcChild);
    rcChild.OffsetRect(ptOffset);
    pwndChild->MoveWindow(rcChild, FALSE);
    pwndChild = pwndChild->GetNextWindow();
} */
```

```
// Adjust the dialog window dimensions
CRect rcWindow;
GetWindowRect(rcWindow);
rcWindow.right += rcClientStart.Width() - rcClientNow.Width();
rcWindow.bottom += rcClientStart.Height() - rcClientNow.Height();
rcWindow.bottom += 5;
MoveWindow(rcWindow, FALSE);
```

```
// And position the control bars
RepositionBars(AFX_IDW_CONTROLBAR_FIRST, AFX_IDW_CONTROLBAR_LAST, 0);
```

```
m_timerID = 0;
m_timerID = SetTimer(TIMERID, 50, NULL);
m_ErrorCounter = 0;
```

```
m_HDCDDisplay = false;
DisplayMode(1);
SetWindowText("Untitled");
partsList[0] = rcClientNow.Width()/4;
partsList[1] = partsList[0] + (rcClientNow.Width()/4);
partsList[2] = partsList[1] + (rcClientNow.Width()/4);
partsList[3] = -1;
m_StatusBar.GetStatusBarCtrl().SetParts(sizeof(partsList)/sizeof(int), partsList);
```

```
return TRUE; // return TRUE unless you set the focus to a control
// EXCEPTION: OCX Property Pages should return FALSE
```

```
}
```

```
void CTabDialog::OnClose()
{
```

```
// if( m_Page1 )
// delete m_Page1;
// if( m_Page2 )
// delete m_Page2;
// if( m_Page3 )
// delete m_Page3;
```

```
if( m_timerID )
    KillTimer( m_timerID );
```

```
delete g_DSPManager;
```

```
CDialog::OnClose();
}
```

```
void CTabDialog::OnSysCommand(UINT nID, LPARAM lParam)
```

```
{
    if ((nID & 0xFFFF) == IDM_ABOUTBOX)
    {
        CTabAboutDlg dlgAbout;
```

0098

```

        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

void CTabDialog::OnPaint()
{
    // CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here

    // Do not call CDialog::OnPaint() for painting messages

    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        //m_StatusBar.RedrawWindow();
        CDialog::OnPaint();
    }
}

void CTabDialog::OnMove(int x, int y)
{
    CDialog::OnMove(x, y);

    // TODO: Add your message handler code here
}

// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CTabDialog::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

HBRUSH CTabDialog::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
    HBRUSH hbr = CDialog::OnCtlColor(pDC, pWnd, nCtlColor);

    // TODO: Change any attributes of the DC here

    // TODO: Return a different brush if the default is not desired
    return hbr;
#ifdef 0
    return m_brush;
#endif
}

void CTabDialog::OnOpenMenu()
{

```

```

// TODO: Add your command handler code here
// char BASED_CODE szFilter[] = "Text Files (*.txt)|*.txt|";
// CFileDialog fileDlg(TRUE,NULL,NULL,0L,szFilter,NULL);

// CFileDialog fileDlg(TRUE);
CFileDialog fileDlg(TRUE,"txt",NULL,OFN_HIDEREADONLY,"Text Files (*.txt)|*.txt|All Files (*.*)|*.*||",this)
if( fileDlg.DoModal() == IDOK )
{
    int array_size = 0;
    CString string;
    CStringArray array;
    CStdioFile file(fileDlg.GetPathName(),CFile::modeRead | CFile::typeText);
    while( file.ReadString(string) )
    {
        array.SetAtGrow(array_size, string );
        array_size++;
    }
    g_DSPManager->SetDSPSettings(array);
    SetWindowText(fileDlg.GetFileName());
    // MM 5/13/99 Invalidate the window so that window is redrawn
    Invalidate(false);
}

}

void CTabDialog::OnSaveMenu()
{
    // TODO: Add your command handler code here

    CFileDialog fileDlg(FALSE,"txt",NULL,OFN_OVERWRITEPROMPT|OFN_HIDEREADONLY,"Text Files (*.txt)|*.txt|",this);

    if( fileDlg.DoModal() == IDOK )
    {
        int array_index = 0;
        CString string;
        CStringArray array;
        g_DSPManager->GetDSPSettings(array);
        CStdioFile file(fileDlg.GetPathName(),CFile::modeCreate | CFile::modeWrite | CFile::typeText);
        while( array_index < array.GetSize() )
        {
            string = array.GetAt(array_index);
            file.WriteString(string);
            file.WriteString("\n");
            array_index++;
        }
    }
}

void CTabDialog::OnFileExportparamfile()
{
    CFileDialog fileDlg(FALSE,"prm",NULL,OFN_OVERWRITEPROMPT|OFN_HIDEREADONLY,"Parameter Files (*.prm)|*.prm|",this);

    if( fileDlg.DoModal() == IDOK )
    {
        int array_index = 0;
        CString string;
        CStringArray array;
        g_DSPManager->GetFilterBlob(array);
        CStdioFile file(fileDlg.GetPathName(),CFile::modeCreate | CFile::modeWrite | CFile::typeBinary);
        while( array_index < array.GetSize() )
        {
            string = array.GetAt(array_index);
            file.WriteString(string);
            file.WriteString("\n");
            array_index++;
        }
    }
}

```

0100

```

    }
}

void CTabDialog::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    if( nIDEvent == TIMERID )
    {
        DisplayMode(g_DSPManager->GetHDCDMode());
        DisplayI2CState(0L);
    }
    else
    {
        CDialog::OnTimer(nIDEvent);
    }
}

void CTabDialog::DisplayMode(int value)
{
    if( value < 0 )
    {
        DisplayI2CState(-value);
        /* char s[100];
        sprintf(s,"0x%x",-value);
        m_StatusBar.GetStatusBarCtrl().SetText(s,3,0); */
        /* if( m_timerID )
        KillTimer( m_timerID );
        CI2CCommErrorDialog dlg;
        char s[100];
        sprintf(s,"0x%x",-value);
        dlg.m_ErrorCode = s;
        if( dlg.DoModal() == IDCANCEL )
        EndDialog(IDCANCEL);
        else
        m_timerID = SetTimer(TIMERID,50,NULL); */
    }
    else if( value )
    {
        if( !m_HDCDDisplay )
        {
            m_ICnt = 0;
            m_AIdx = 0;
            m_StatusBar.GetStatusBarCtrl().SetText(Detect_Array[m_AIdx++],1,0);
            //SetWindowText(Detect_Array[m_AIdx++]);
        }
        else
        {
            m_ICnt++;
            if( m_ICnt > 5 )
            {
                m_ICnt = 0;
                m_StatusBar.GetStatusBarCtrl().SetText(Detect_Array[m_AIdx++],1,0);
                //SetWindowText(Detect_Array[m_AIdx++]);
                if( m_AIdx >= AARRAYSIZE )
                    m_AIdx = 0;
            }
        }
        m_HDCDDisplay = true;
    }
    else
    {
        if( m_HDCDDisplay )
        {
            m_StatusBar.GetStatusBarCtrl().SetText("BAD CD..NO DONUT",1,0);
            //SetWindowText("BAD CD .. NO DONUT");
            m_HDCDDisplay = false;
        }
    }
}

```

0101



```

)

void CTabDialog::SetStatusString(int which, CString & s)
{
    m_StatusBar.GetStatusBarCtrl().SetText(s,0,0);
}

void CTabDialog::DisplayI2CState(long error)
{
    if( error )
    {
        char s[100];
        sprintf(s,"0x%x",error);
        m_StatusBar.GetStatusBarCtrl().SetText(s,3,0);
        m_ErrorCounter = 100;    // 5 seconds
    }
    else
    {
        if( m_ErrorCounter )
        {
            m_ErrorCounter--;
            if( m_ErrorCounter > 0 )
                return;
        }
        if( g_DSPManager->IsBusy() )
        {
            m_StatusBar.GetStatusBarCtrl().SetText("Transmitting..",3,0);
        }
        else
        {
            m_StatusBar.GetStatusBarCtrl().SetText("",3,0);
        }
    }
}
}

```

```

#if !defined(AFX_STWAVEREJPAGE_H__0CF7E209_D790_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_STWAVEREJPAGE_H__0CF7E209_D790_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// StWaveRejPage.h : header file
//

////////////////////////////////////
// CStWaveRejPage dialog

class CTabDialog;

class CStWaveRejPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CStWaveRejPage)

// Construction
public:
    CStWaveRejPage();
    ~CStWaveRejPage();

// Dialog Data
    //{{AFX_DATA(CStWaveRejPage)
    enum { IDD = IDD_PP5 };
    CButton m_BypassButton;
    CSliderCtrl m_Notch3BoostSlider;
    CSliderCtrl m_Notch3QSlider;
    CSliderCtrl m_Notch2CutSlider;
    CSliderCtrl m_Notch2QSlider;
    CSliderCtrl m_Notch2FreqSlider;
    CSliderCtrl m_Notch1CutSlider;
    CSliderCtrl m_Notch1QSlider;
    CSliderCtrl m_Notch1FreqSlider;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CStWaveRejPage)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CStWaveRejPage)
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
    afx_msg void OnPaint();
    afx_msg void OnBypass();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_STWAVEREJPAGE_H__0CF7E209_D790_11D2_96EE_006097CDB9E2__INCLUDED_)

```

```

// StWaveRejPage.cpp : Implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "StWaveRejPage.h"
#include "TabDialog.h"
#include "DSP56kManager.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CStWaveRejPage property page

IMPLEMENT_DYNCREATE(CStWaveRejPage, CPropertyPage)

CStWaveRejPage::CStWaveRejPage() : CPropertyPage(CStWaveRejPage::IDD)
{
    //{{AFX_DATA_INIT(CStWaveRejPage)
    //}}AFX_DATA_INIT
}

CStWaveRejPage::~CStWaveRejPage()
{
}

void CStWaveRejPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CStWaveRejPage)
    DDX_Control(pDX, IDC_CHECK5, m_BypassButton);
    DDX_Control(pDX, IDC_SLIDER9, m_Notch3BoostSlider);
    DDX_Control(pDX, IDC_SLIDER8, m_Notch3QSlider);
    DDX_Control(pDX, IDC_SLIDER6, m_Notch2CutSlider);
    DDX_Control(pDX, IDC_SLIDER5, m_Notch2QSlider);
    DDX_Control(pDX, IDC_SLIDER4, m_Notch2FreqSlider);
    DDX_Control(pDX, IDC_SLIDER3, m_Notch1CutSlider);
    DDX_Control(pDX, IDC_SLIDER2, m_Notch1QSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_Notch1FreqSlider);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CStWaveRejPage, CPropertyPage)
    //{{AFX_MSG_MAP(CStWaveRejPage)
    ON_WM_VSCROLL()
    ON_WM_SHOWWINDOW()
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_CHECK5, OnBypass)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CStWaveRejPage message handlers

BOOL CStWaveRejPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_Notch1FreqSlider.SetRange(0, CONTROL_RANGE);
    m_Notch1FreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch3Freq));
    m_Notch1FreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_Notch1FreqSlider.SetPageSize(PG_CONTROL_AMT);

    m_Notch1QSlider.SetRange(0, CONTROL_RANGE);
    m_Notch1QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch3Q));
    m_Notch1QSlider.SetTicFreq((CONTROL_RANGE+1)/16);

```

```

m_Notch1QSlider.SetPage(PG_CONTROL_AMT);

m_Notch1CutSlider.SetRange(0,CONTROL_RANGE);
m_Notch1CutSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch3Cut));
m_Notch1CutSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_Notch1CutSlider.SetPageSize(PG_CONTROL_AMT);

m_Notch2FreqSlider.SetRange(0,CONTROL_RANGE);
m_Notch2FreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch4Freq));
m_Notch2FreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_Notch2FreqSlider.SetPageSize(PG_CONTROL_AMT);

m_Notch2QSlider.SetRange(0,CONTROL_RANGE);
m_Notch2QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch4Q));
m_Notch2QSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_Notch2QSlider.SetPageSize(PG_CONTROL_AMT);

m_Notch2CutSlider.SetRange(0,CONTROL_RANGE);
m_Notch2CutSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch4Cut));
m_Notch2CutSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_Notch2CutSlider.SetPageSize(PG_CONTROL_AMT);

m_Notch3QSlider.SetRange(0,CONTROL_RANGE);
m_Notch3QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch5Q));
m_Notch3QSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_Notch3QSlider.SetPageSize(PG_CONTROL_AMT);

m_Notch3BoostSlider.SetRange(0,CONTROL_RANGE);
m_Notch3BoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch5Cut));
m_Notch3BoostSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_Notch3BoostSlider.SetPageSize(PG_CONTROL_AMT);

m_ParentWindow = (CDialog *) GetParent()->GetParent();

return TRUE; // return TRUE unless you set the focus to a control
// EXCEPTION: OCK Property Pages should return FALSE
}

void CStWaveRejPage::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which,str);
    m_ParentWindow->SetStatusString(0,str);
}

void CStWaveRejPage::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if( slider == &m_Notch1FreqSlider )
        which = kNotch3Freq;
    else if( slider == &m_Notch1QSlider )
        which = kNotch3Q;
    else if( slider == &m_Notch1CutSlider )
        which = kNotch3Cut;
    else if( slider == &m_Notch2FreqSlider )
        which = kNotch4Freq;
    else if( slider == &m_Notch2QSlider )
        which = kNotch4Q;
    else if( slider == &m_Notch2CutSlider )
        which = kNotch4Cut;
    else if( slider == &m_Notch3QSlider )
        which = kNotch5Q;

```

0105

```

else if( slider == &m_Th3BoostSlider )
    which = kNotch5Cut;
else
    return;

g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(), which);
SendStringToUI(which);
}

void CStWaveRejPage::OnShowWindow(BOOL bShow, UINT nStatus)
{
    CPropertyPage::OnShowWindow(bShow, nStatus);

    // TODO: Add your message handler code here
}

void CStWaveRejPage::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_Notch1FreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch3Freq));
    m_Notch1QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch3Q));
    m_Notch1CutSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch3Cut));

    m_Notch2FreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch4Freq));
    m_Notch2QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch4Q));
    m_Notch2CutSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch4Cut));

    m_Notch3QSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch5Q));
    m_Notch3BoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kNotch5Cut));

    m_BypassButton.SetCheck(g_DSPManager->GetBypassSection(kBypassConecry));
    // Do not call CPropertyPage::OnPaint() for painting messages
}

void CStWaveRejPage::OnBypass()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassConecry);
}

```

/\*

There a number of blocks (processing units).  
The signal flow is linear currently.  
Each item is an ascii line of text  
The file format is as follows:

version number

block type  
number of params for block  
block param #1  
block param #2  
block param #3  
...  
block param #n

block type  
number of params for block  
block param #1  
block param #2  
block param #3  
...  
block param #n

...

\*/

// file version number  
#define BLOB\_FILE\_VERSION 1

// block type descriptor  
typedef enum {  
 BLOCK\_DELAY,  
 BLOCK\_GAIN,  
 BLOCK\_PEQ,  
 BLOCK\_AP,  
 BLOCK\_LS,  
 BLOCK\_HS,  
 BLOCK\_HP,  
 BLOCK\_LP,  
 BLOCK\_LP2  
} ProcessType;

0107

```

// SPIPEMicroComm.h: interface for the SPIPEMicroComm class.
//
////////////////////////////////////
#if !defined(AFX_SPIPEMICROCOMM_H__33F9EF06_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_SPIPEMICROCOMM_H__33F9EF06_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "DSPComm.h"

class SPIPEMicroComm : public DSPComm
{
    int cmd_delay;
    int io_delay;

public:
    SPIPEMicroComm();
    virtual ~SPIPEMicroComm();
    virtual long SendDSPWord(long);
    virtual long SendDSPMemory(char *, long);
};

#endif // !defined(AFX_SPIPEMICROCOMM_H__33F9EF06_F6EF_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0108

```

// SPIPEMicroComm.cpp: implementation of the SPIPEMicroComm class.
//
/////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "sa.h"
#include "SPIPEMicroComm.h"

#include "unit_ppi.h"
#include "functs.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

/////////////////////////////////////////////////////////////////
// Construction/Destruction
/////////////////////////////////////////////////////////////////

SPIPEMicroComm::SPIPEMicroComm()
{
    cmd_delay = 1000;
    io_delay = 50;

    if( load_dll()==false )
    {
        MessageBox(NULL, "UNIT_PPI not found in the system directory. Fix this and restart application.", "Danger, I
anger Will Robinson", MB_OK | MB_ICONSTOP );
    }
    else
    {
        init_port_spi(1);
        set_cmd_delay_cnt_value(cmd_delay);
        set_io_delay_cnt_value(io_delay);
    }
}

SPIPEMicroComm::~SPIPEMicroComm()
{
}

long SPIPEMicroComm::SendDSPWord(long value)
{
    return( spi_xchnge24(value) );
}

long SPIPEMicroComm::SendDSPMemory(char *data, long len)
{
    return( 0L );
}

```



```

#if !defined(AFX_SHELVPA..._6F151624_D84D_11D2_96EE_006097CDB9E2__INCLUDED_)
#define AFX_SHELVPA..._6F151624_D84D_11D2_96EE_006097CDB9E2__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// ShelvPage.h : header file
//

////////////////////////////////////

// CShevPage dialog

class CTabDialog;

class CShevPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CShevPage)

// Construction
public:
    CShevPage();
    ~CShevPage();

// Dialog Data
    //{{AFX_DATA(CShevPage)
    enum { IDD = IDD_PP6 };
    CButton m_BypassSecondButton;
    CButton m_BypassFirstButton;
    CSliderCtrl m_HiBoostSlider;
    CSliderCtrl m_HiFreqSlider;
    CSliderCtrl m_LoBoostSlider;
    CSliderCtrl m_LoFreqSlider;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CShevPage)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CShevPage)
    virtual BOOL OnInitDialog();
    afx_msg void OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    afx_msg void OnShowWindow(BOOL bShow, UINT nStatus);
    afx_msg void OnPaint();
    afx_msg void OnBypassFirst();
    afx_msg void OnBypassSecond();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

    CTabDialog *m_ParentWindow;
    void SendStringToUI(int which);

};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_SHELVPA..._6F151624_D84D_11D2_96EE_006097CDB9E2__INCLUDED_)

```

0110

```

// ShelvPage.cpp : implementation file
//

#include "stdafx.h"
#include "sa.h"
#include "ShelvPage.h"
#include "TabDialog.h"
#include "DSP56kManager.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CShelvPage property page

IMPLEMENT_DYNCREATE(CShelvPage, CPropertyPage)

CShelvPage::CShelvPage() : CPropertyPage(CShelvPage::IDD)
{
    //{{AFX_DATA_INIT(CShelvPage)
    //}}AFX_DATA_INIT
}

CShelvPage::~CShelvPage()
{
}

void CShelvPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CShelvPage)
    DDX_Control(pDX, IDC_CHECK3, m_BypassSecondButton);
    DDX_Control(pDX, IDC_CHECK1, m_BypassFirstButton);
    DDX_Control(pDX, IDC_SLIDER6, m_HiBoostSlider);
    DDX_Control(pDX, IDC_SLIDER4, m_HiFreqSlider);
    DDX_Control(pDX, IDC_SLIDER3, m_LoBoostSlider);
    DDX_Control(pDX, IDC_SLIDER1, m_LoFreqSlider);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CShelvPage, CPropertyPage)
    //{{AFX_MSG_MAP(CShelvPage)
    ON_WM_VSCROLL()
    ON_WM_SHOWWINDOW()
    ON_WM_PAINT()
    ON_BN_CLICKED(IDC_CHECK1, OnBypassFirst)
    ON_BN_CLICKED(IDC_CHECK3, OnBypassSecond)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CShelvPage message handlers

BOOL CShelvPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    // TODO: Add extra initialization here
    m_LoFreqSlider.SetRange(0, CONTROL_RANGE);
    m_LoFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoShelfFreq));
    m_LoFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_LoFreqSlider.SetPageSize(PG_CONTROL_AMT);

    m_LoBoostSlider.SetRange(0, CONTROL_RANGE);
    m_LoBoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoShelfGain));
    m_LoBoostSlider.SetTicFreq((CONTROL_RANGE+1)/16);
    m_LoBoostSlider.SetPageSize(PG_CONTROL_AMT);
}

```

0111

```

m_HiFreqSlider.SetRange(CONTROL_RANGE);
m_HiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiShelfFreq));
m_HiFreqSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_HiFreqSlider.SetPageSize(PG_CONTROL_AMT);

m_HiBoostSlider.SetRange(0,CONTROL_RANGE);
m_HiBoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiShelfGain));
m_HiBoostSlider.SetTicFreq((CONTROL_RANGE+1)/16);
m_HiBoostSlider.SetPageSize(PG_CONTROL_AMT);

m_ParentWindow = (CTabDialog *) GetParent()->GetParent();

return TRUE; // return TRUE unless you set the focus to a control
            // EXCEPTION: OCX Property Pages should return FALSE
}

void CShelvPage::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    // TODO: Add your message handler code here and/or call default
    CSliderCtrl *slider = (CSliderCtrl *)pScrollBar;
    int which;

    CPropertyPage::OnVScroll(nSBCode, nPos, pScrollBar);

    Sleep(50);
    if( slider == &m_LoFreqSlider )
        which = kLoShelfFreq;
    else if( slider == &m_LoBoostSlider )
        which = kLoShelfGain;
    else if( slider == &m_HiFreqSlider )
        which = kHiShelfFreq;
    else if( slider == &m_HiBoostSlider )
        which = kHiShelfGain;
    else
        return;

    g_DSPManager->SetParamValue(CONTROL_RANGE-slider->GetPos(),which);
    SendStringToUI(which);
}

void CShelvPage::SendStringToUI(int which)
{
    CString str;

    g_DSPManager->GetStringValue(which,str);
    m_ParentWindow->SetStatusString(0,str);
}

void CShelvPage::OnShowWindow(BOOL bShow, UINT nStatus)
{
    CPropertyPage::OnShowWindow(bShow, nStatus);

    // TODO: Add your message handler code here
}

void CShelvPage::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    m_LoFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoShelfFreq));
    m_LoBoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kLoShelfGain));
    m_HiFreqSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiShelfFreq));
    m_HiBoostSlider.SetPos(CONTROL_RANGE-g_DSPManager->GetParamValue(kHiShelfGain));
    m_BypassFirstButton.SetCheck(g_DSPManager->GetBypassSection(kBypassLoshelf));
}

```

```

    m_BypassSecondButton.SendMessage(WM_COMMAND, IDBYPASS, 0);
    // Do not call CPropertyPage::OnPaint() for painting messages
}

void CShelvPage::OnBypassFirst()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassFirstButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassLoShelf);
}

void CShelvPage::OnBypassSecond()
{
    // TODO: Add your control notification handler code here
    int state = m_BypassSecondButton.GetState() & 0x3;

    g_DSPManager->SetBypassSection(state, kBypassHiShelf);
}

```

```

//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif // _WIN32

#ifdef APSTUDIO_INVOKED
////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include \"afxres.h\"\\r\\n"
    "\\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\\r\\n"
    "#define _AFX_NO_OLE_RESOURCES\\r\\n"
    "#define _AFX_NO_TRACKER_RESOURCES\\r\\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\\r\\n"
    "\\r\\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\\r\\n"
    "#ifdef _WIN32\\r\\n"
    "LANGUAGE 9, 1\\r\\n"
    "#pragma code_page(1252)\\r\\n"
    "#endif\\r\\n"
    "#include \"res\\sa.rc2\" // non-Microsoft Visual C++ edited resources\\r\\n"
    "#include \"afxres.rc\" // Standard components\\r\\n"
    "#endif\\0"
END

#endif // APSTUDIO_INVOKED

////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME ICON DISCARDABLE "res\\sa.ico"

////////////////////////////////////
//
// Dialog

```

```

IDD_ABOUTBOX DIALOG DISCARDABLE 0, 0, 259, 55
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About KOJ's Amazing Town Crier"
FONT 8, "MS Sans Serif"
BEGIN
    ICON                IDR_MAINFRAME, IDC_STATIC, 11, 17, 21, 20
    LTEXT                "KOJ's Amazing Town Crier Version 1.0", IDC_STATIC, 40, 10,
                        119, 8, SS_NOPREFIX
    LTEXT                "Copyright (C) PMI 1999", IDC_STATIC, 40, 25, 119, 8
    DEFPUSHBUTTON        "OK", IDOK, 220, 7, 32, 14, WS_GROUP
END

```

```

IDD_SA_DIALOG DIALOGEX 0, 0, 229, 175
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "SuperAudio"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON        "OK", IDOK, 172, 7, 50, 14
    PUSHBUTTON           "Cancel", IDCANCEL, 172, 23, 50, 14
    PUSHBUTTON           "Poke Sequence #1", IDC_BUTTON1, 35, 83, 109, 25
    PUSHBUTTON           "Poke Sequence #2", IDC_BUTTON2, 33, 123, 115, 28
END

```

```

IDD_DIALOG1 DIALOG DISCARDABLE 0, 0, 236, 229
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "KOJ's Amazing Town Crier"
MENU IDR_MENU1
FONT 8, "MS Sans Serif"
BEGIN
END

```

```

IDD_PP1 DIALOG DISCARDABLE 0, 0, 195, 127
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Test"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON           "Send SPI Sequence #1", IDC_BUTTON1, 57, 36, 86, 22
    PUSHBUTTON           "Send SPI Sequence #2", IDC_BUTTON2, 55, 81, 93, 31
END

```

```

IDD_PP2 DIALOG DISCARDABLE 0, 0, 288, 146
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Main"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL              "Slider1", IDC_SLIDER1, "msctls_trackbar32", TBS_AUTOTICKS |
                        TBS_VERT | WS_TABSTOP, 58, 25, 22, 86
    LTEXT                "Pre", IDC_STATIC, 28, 117, 12, 8
    CONTROL              "Bypass Processing", IDC_CHECK1, "Button", BS_AUTOCHECKBOX |
                        WS_TABSTOP, 173, 57, 76, 10
    CONTROL              "Slider1", IDC_SLIDER3, "msctls_trackbar32", TBS_AUTOTICKS |
                        TBS_VERT | WS_TABSTOP, 27, 25, 22, 86
    LTEXT                "Post", IDC_STATIC, 57, 117, 15, 8
    GROUPBOX             "Volume", IDC_STATIC, 19, 15, 135, 118
    CONTROL              "Bypass HDCD", IDC_CHECK2, "Button", BS_AUTOCHECKBOX |
                        WS_TABSTOP, 173, 74, 76, 10
    CONTROL              "Bypass HDCD Gain Scale", IDC_CHECK3, "Button",
                        BS_AUTOCHECKBOX | WS_TABSTOP, 173, 91, 98, 10
    PUSHBUTTON           "Reset All", IDC_BUTTON1, 173, 31, 50, 14
    CONTROL              "Analog Input", IDC_CHECK4, "Button", BS_AUTOCHECKBOX |
                        WS_TABSTOP, 173, 108, 56, 10
    CONTROL              "Slider1", IDC_SLIDER7, "msctls_trackbar32", TBS_AUTOTICKS |
                        TBS_VERT | WS_TABSTOP, 89, 25, 22, 86
    LTEXT                "Analog", IDC_STATIC, 85, 117, 23, 8
    CONTROL              "Slider1", IDC_SLIDER8, "msctls_trackbar32", TBS_AUTOTICKS |
                        TBS_VERT | WS_TABSTOP, 119, 25, 22, 86
    LTEXT                "Delay", IDC_STATIC, 117, 117, 19, 8
END

```

```

IDD_PP3 DIALOG DISCARDABLE 0, 0, 195, 127
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Compression"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL                "Bypass Compression", IDC_CHECK1, "Button", BS_AUTOCHECKBOX |
                           WS_TABSTOP, 29, 24, 104, 10
END

```

```

IDD_PP4 DIALOG DISCARDABLE 0, 0, 244, 159
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Resonance Compensation"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL                "Slider1", IDC_SLIDER1, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 20, 24, 20, 85
    CONTROL                "Slider1", IDC_SLIDER2, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 48, 24, 21, 85
    CONTROL                "Slider1", IDC_SLIDER3, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 76, 24, 20, 85
    LTEXT                  "Freq.", IDC_STATIC, 17, 113, 20, 11
    LTEXT                  "Q", IDC_STATIC, 51, 113, 8, 11
    LTEXT                  "Cut/Boost", IDC_STATIC, 71, 113, 33, 11
    GROUPBOX               "First Notch", IDC_STATIC, 11, 15, 98, 111
    CONTROL                "Slider1", IDC_SLIDER4, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 137, 24, 24, 85
    CONTROL                "Slider1", IDC_SLIDER5, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 165, 24, 21, 85
    CONTROL                "Slider1", IDC_SLIDER6, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 193, 24, 22, 85
    LTEXT                  "Freq.", IDC_STATIC, 135, 113, 20, 11
    LTEXT                  "Q", IDC_STATIC, 169, 113, 8, 11
    LTEXT                  "Cut/Boost", IDC_STATIC, 189, 113, 32, 11
    GROUPBOX               "Second Notch", IDC_STATIC, 129, 15, 98, 111
    CONTROL                "Bypass", IDC_CHECK5, "Button", BS_AUTOCHECKBOX |
                           WS_TABSTOP, 33, 136, 39, 10
    CONTROL                "Bypass", IDC_CHECK6, "Button", BS_AUTOCHECKBOX |
                           WS_TABSTOP, 153, 136, 39, 10
END

```

```

IDD_PP5 DIALOG DISCARDABLE 0, 0, 339, 162
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Standing Wave Rejection"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL                "Slider1", IDC_SLIDER1, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 20, 24, 20, 85
    CONTROL                "Slider1", IDC_SLIDER2, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 48, 24, 21, 85
    CONTROL                "Slider1", IDC_SLIDER3, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 76, 24, 20, 85
    LTEXT                  "Freq.", IDC_STATIC, 17, 113, 20, 11
    LTEXT                  "Q", IDC_STATIC, 51, 113, 8, 11
    LTEXT                  "Cut/Boost", IDC_STATIC, 70, 113, 35, 11
    GROUPBOX               "First Notch", IDC_STATIC, 11, 15, 98, 111
    CONTROL                "Slider1", IDC_SLIDER4, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 229, 24, 24, 85
    CONTROL                "Slider1", IDC_SLIDER5, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 257, 24, 21, 85
    CONTROL                "Slider1", IDC_SLIDER6, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 285, 24, 22, 85
    LTEXT                  "Freq.", IDC_STATIC, 227, 113, 20, 11
    LTEXT                  "Q", IDC_STATIC, 261, 113, 8, 11
    LTEXT                  "Cut/Boost", IDC_STATIC, 280, 113, 34, 11
    GROUPBOX               "Second Notch", IDC_STATIC, 221, 15, 98, 111
    CONTROL                "Slider1", IDC_SLIDER8, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 140, 25, 21, 85
    CONTROL                "Slider1", IDC_SLIDER9, "msctls_trackbar32", TBS_AUTOTICKS |
                           TBS_VERT | WS_TABSTOP, 168, 25, 22, 85
    LTEXT                  "Q", IDC_STATIC, 144, 114, 8, 11
    LTEXT                  "Cut/Boost", IDC_STATIC, 163, 114, 34, 11
    GROUPBOX               "Boost Compensation", IDC_STATIC, 125, 15, 77, 111

```

END  
"BYPASS", IDC\_CHECK5, "Button", BS\_AUTOCHECKBOX |  
WS\_TABSTOP, 146, 137, 41, 10

IDD\_PP6 DIALOG DISCARDABLE 0, 0, 187, 162  
STYLE WS\_CHILD | WS\_DISABLED | WS\_CAPTION  
CAPTION "Smiley-Face"  
FONT 8, "MS Sans Serif"  
BEGIN

CONTROL "Slider1", IDC\_SLIDER1, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 20, 24, 20, 85  
CONTROL "Slider1", IDC\_SLIDER3, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 50, 24, 20, 85  
LTEXT "Freq.", -1, 17, 113, 20, 11  
LTEXT "Cut/Boost", -1, 43, 113, 34, 11  
GROUPBOX "Lo-Shelf", -1, 11, 15, 70, 111  
CONTROL "Slider1", IDC\_SLIDER4, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 108, 24, 24, 85  
CONTROL "Slider1", IDC\_SLIDER6, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 139, 24, 22, 85  
LTEXT "Freq.", -1, 106, 113, 20, 11  
LTEXT "Cut/Boost", -1, 134, 113, 33, 11  
GROUPBOX "Hi-Shelf", -1, 100, 15, 69, 111  
CONTROL "Bypass", IDC\_CHECK1, "Button", BS\_AUTOCHECKBOX |  
WS\_TABSTOP, 25, 136, 37, 12  
CONTROL "Bypass", IDC\_CHECK3, "Button", BS\_AUTOCHECKBOX |  
WS\_TABSTOP, 115, 136, 37, 12

END

IDD\_PP7 DIALOG DISCARDABLE 0, 0, 242, 162  
STYLE WS\_CHILD | WS\_DISABLED | WS\_CAPTION  
CAPTION "Cutoff Response"  
FONT 8, "MS Sans Serif"  
BEGIN

CONTROL "Slider1", IDC\_SLIDER1, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 20, 24, 20, 85  
CONTROL "Slider1", IDC\_SLIDER2, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 48, 24, 21, 85  
CONTROL "Slider1", IDC\_SLIDER3, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 76, 24, 20, 85  
LTEXT "Freq.", IDC\_STATIC, 17, 113, 20, 11  
LTEXT "Q", IDC\_STATIC, 51, 113, 8, 11  
LTEXT "Cut/Boost", IDC\_STATIC, 71, 113, 33, 11  
GROUPBOX "Low Cutoff", IDC\_STATIC, 11, 15, 98, 111  
CONTROL "Slider1", IDC\_SLIDER4, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 137, 24, 24, 85  
CONTROL "Slider1", IDC\_SLIDER5, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 165, 24, 21, 85  
CONTROL "Slider1", IDC\_SLIDER6, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 193, 24, 22, 85  
LTEXT "Freq.", IDC\_STATIC, 135, 113, 20, 11  
LTEXT "Q", IDC\_STATIC, 169, 113, 8, 11  
LTEXT "Cut/Boost", IDC\_STATIC, 189, 113, 32, 11  
GROUPBOX "High Cutoff", IDC\_STATIC, 129, 15, 98, 111  
CONTROL "Bypass", IDC\_CHECK5, "Button", BS\_AUTOCHECKBOX |  
WS\_TABSTOP, 39, 139, 41, 12  
CONTROL "Bypass", IDC\_CHECK6, "Button", BS\_AUTOCHECKBOX |  
WS\_TABSTOP, 154, 139, 41, 12

END

IDD\_PP8 DIALOG DISCARDABLE 0, 0, 285, 164  
STYLE WS\_CHILD | WS\_DISABLED | WS\_CAPTION  
CAPTION "Cutoff Response"  
FONT 8, "MS Sans Serif"  
BEGIN

CONTROL "Slider1", IDC\_SLIDER1, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 20, 24, 20, 85  
CONTROL "Slider1", IDC\_SLIDER2, "msctls\_trackbar32", TBS\_AUTOTICKS |  
TBS\_VERT | WS\_TABSTOP, 48, 24, 21, 85  
LTEXT "Freq.", -1, 17, 113, 20, 11  
LTEXT "Q", -1, 51, 113, 8, 11  
GROUPBOX "Low Cutoff", -1, 11, 15, 71, 111

0117



```

CONTROL      "Slider1", IDC_SLIDER4, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 112, 24, 24, 85
CONTROL      "Slider1", IDC_SLIDER5, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 140, 24, 21, 85
LTEXT        "Freq.", -1, 110, 113, 20, 11
LTEXT        "Q", -1, 144, 113, 8, 11
GROUPBOX     "High Cutoff", -1, 104, 15, 69, 111
CONTROL      "Bypass", IDC_CHECK4, "Button", BS_AUTOCHECKBOX |
WS_TABSTOP, 25, 136, 40, 13
CONTROL      "Bypass", IDC_CHECK5, "Button", BS_AUTOCHECKBOX |
WS_TABSTOP, 116, 136, 40, 13
CONTROL      "Slider1", IDC_SLIDER8, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 203, 25, 24, 85
CONTROL      "Slider1", IDC_SLIDER9, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 231, 25, 21, 85
LTEXT        "Freq.", -1, 201, 114, 20, 11
LTEXT        "Q", -1, 235, 114, 8, 11
GROUPBOX     "New High Cutoff", -1, 195, 15, 69, 111
CONTROL      "Bypass", IDC_CHECK7, "Button", BS_AUTOCHECKBOX |
WS_TABSTOP, 207, 137, 40, 13
END

```

```

IDD_PP9 DIALOG DISCARDABLE 0, 0, 244, 164
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Resonance Compensation #2"
FONT 8, "MS Sans Serif"
BEGIN

```

```

CONTROL      "Slider1", IDC_SLIDER1, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 20, 24, 20, 85
CONTROL      "Slider1", IDC_SLIDER2, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 48, 24, 21, 85
CONTROL      "Slider1", IDC_SLIDER3, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 76, 24, 20, 85
LTEXT        "Freq.", IDC_STATIC, 17, 113, 20, 11
LTEXT        "Q", IDC_STATIC, 51, 113, 8, 11
LTEXT        "Cut/Boost", IDC_STATIC, 71, 113, 33, 11
GROUPBOX     "Third Notch", IDC_STATIC, 11, 15, 98, 111
CONTROL      "Slider1", IDC_SLIDER4, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 137, 24, 24, 85
CONTROL      "Slider1", IDC_SLIDER5, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 165, 24, 21, 85
CONTROL      "Slider1", IDC_SLIDER6, "msctls_trackbar32", TBS_AUTOTICKS |
TBS_VERT | WS_TABSTOP, 193, 24, 22, 85
LTEXT        "Freq.", IDC_STATIC, 135, 113, 20, 11
LTEXT        "Q", IDC_STATIC, 169, 113, 8, 11
LTEXT        "Cut/Boost", IDC_STATIC, 189, 113, 32, 11
GROUPBOX     "Fourth Notch", IDC_STATIC, 129, 15, 98, 111
CONTROL      "Bypass", IDC_CHECK5, "Button", BS_AUTOCHECKBOX |
WS_TABSTOP, 36, 139, 40, 12
CONTROL      "Bypass", IDC_CHECK6, "Button", BS_AUTOCHECKBOX |
WS_TABSTOP, 155, 139, 40, 12
END

```

```

IDD_DIALOG2 DIALOG DISCARDABLE 0, 0, 175, 66
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN

```

```

PUSHBUTTON   "Cancel", IDCANCEL, 59, 34, 50, 14
LTEXT        "Initializing I2C. Please Wait.....", IDC_STATIC, 43, 17,
98, 8
END

```

```

IDD_DIALOG3 DIALOG DISCARDABLE 0, 0, 186, 132
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN

```

```

DEFPUSHBUTTON "OK", IDOK, 107, 95, 50, 14
CONTROL      "COM1", IDC_RADIO1, "Button", BS_AUTORADIOBUTTON, 23, 57, 36,
10
CONTROL      "COM2", IDC_RADIO2, "Button", BS_AUTORADIOBUTTON, 23, 70, 36,

```

```

CONTROL      10
              "COM3", IDC_RADIO3, "Button", BS_AUTORADIOBUTTON, 23, 82, 36,
              10
CONTROL      10
              "COM4", IDC_RADIO4, "Button", BS_AUTORADIOBUTTON, 23, 95, 36,
              10
LTEXT        "Choose COM port and reset DSP board", IDC_STATIC, 28, 21,
              125, 8
GROUPBOX     "COM Port", IDC_STATIC, 15, 45, 68, 69
END

```

```

IDD_DIALOG4 DIALOG DISCARDABLE  0, 0, 186, 121
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON  "OK", IDOK, 27, 72, 50, 14
    PUSHBUTTON     "Quit App", IDC_BUTTON1, 101, 72, 48, 14
    LTEXT          "I2C communication error!", IDC_STATIC, 49, 25, 79, 8
    EDITTEXT       IDC_EDIT1, 49, 42, 68, 12, ES_AUTOHSCROLL
END

```

```

IDD_PP10 DIALOG DISCARDABLE  0, 0, 187, 162
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Allpass"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL      "Slider1", IDC_SLIDER1, "msctls_trackbar32", TBS_AUTOTICKS |
                  TBS_VERT | WS_TABSTOP, 20, 24, 20, 85
    CONTROL      "Slider1", IDC_SLIDER3, "msctls_trackbar32", TBS_AUTOTICKS |
                  TBS_VERT | WS_TABSTOP, 50, 24, 20, 85
    LTEXT        "Freq.", -1, 17, 113, 20, 11
    LTEXT        "Q", -1, 53, 113, 8, 8
    GROUPBOX     "Allpass", -1, 11, 15, 70, 111
    CONTROL      "Bypass", IDC_CHECK1, "Button", BS_AUTOCHECKBOX |
                  WS_TABSTOP, 25, 136, 37, 12
END

```

```

IDD_PP11 DIALOG DISCARDABLE  0, 0, 244, 159
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Double-Tuned Notch"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL      "Slider1", IDC_SLIDER1, "msctls_trackbar32", TBS_AUTOTICKS |
                  TBS_VERT | WS_TABSTOP, 20, 24, 20, 85
    CONTROL      "Slider1", IDC_SLIDER2, "msctls_trackbar32", TBS_AUTOTICKS |
                  TBS_VERT | WS_TABSTOP, 48, 24, 21, 85
    LTEXT        "Freq.", -1, 17, 113, 20, 11
    LTEXT        "Q", -1, 51, 113, 8, 11
    GROUPBOX     "Notch", -1, 11, 15, 70, 111
    CONTROL      "Slider1", IDC_SLIDER5, "msctls_trackbar32", TBS_AUTOTICKS |
                  TBS_VERT | WS_TABSTOP, 109, 24, 21, 85
    CONTROL      "Slider1", IDC_SLIDER6, "msctls_trackbar32", TBS_AUTOTICKS |
                  TBS_VERT | WS_TABSTOP, 137, 24, 22, 85
    LTEXT        "Q", -1, 113, 113, 8, 11
    LTEXT        "Cut/Boost", -1, 133, 113, 32, 11
    GROUPBOX     "Compensation", -1, 99, 15, 71, 111
    CONTROL      "Bypass", IDC_CHECK5, "Button", BS_AUTOCHECKBOX |
                  WS_TABSTOP, 73, 136, 39, 10
END

```

```

#ifndef _MAC
////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
FILEVERSION 0,7,4,0
PRODUCTVERSION 0,7,4,0
FILEFLAGSMASK 0x3fL
#ifdef _DEBUG

```

0119

```

FILEFLAGS 0x1L
#else
FILEFLAGS 0x0L
#endif
FILEOS 0x4L
FILETYPE 0x1L
FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "CompanyName", "Pacific Microsonics Inc.\0"
            VALUE "FileDescription", "sa MFC Application\0"
            VALUE "FileVersion", "0, 7, 4, 0\0"
            VALUE "InternalName", "sa\0"
            VALUE "LegalCopyright", "Copyright (C) PMI 1999\0"
            VALUE "OriginalFilename", "sa.EXE\0"
            VALUE "ProductName", "sa Application\0"
            VALUE "ProductVersion", "0, 7, 4, 0\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif // !_MAC

```

```

////////////////////////////////////
//
// DESIGNINFO
//

```

```

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN

```

```

    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 252
        TOPMARGIN, 7
        BOTTOMMARGIN, 48
    END

```

```

    IDD_SA_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 222
        TOPMARGIN, 7
        BOTTOMMARGIN, 168
    END

```

```

    IDD_DIALOG1, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 229
        TOPMARGIN, 7
        BOTTOMMARGIN, 222
    END

```

```

    IDD_PP1, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 188
        TOPMARGIN, 7
        BOTTOMMARGIN, 120
    END

```

```

    IDD_PP2, DIALOG

```

```

BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 281
  TOPMARGIN, 7
  BOTTOMMARGIN, 139
END

IDD_PP3, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 188
  TOPMARGIN, 7
  BOTTOMMARGIN, 120
END

IDD_PP4, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 237
  TOPMARGIN, 7
  BOTTOMMARGIN, 152
END

IDD_PP5, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 332
  TOPMARGIN, 7
  BOTTOMMARGIN, 155
END

IDD_PP6, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 180
  TOPMARGIN, 7
  BOTTOMMARGIN, 155
END

IDD_PP7, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 235
  TOPMARGIN, 7
  BOTTOMMARGIN, 155
END

IDD_PP8, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 278
  TOPMARGIN, 7
  BOTTOMMARGIN, 157
END

IDD_PP9, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 237
  TOPMARGIN, 7
  BOTTOMMARGIN, 157
END

IDD_DIALOG2, DIALOG
BEGIN
  LEFTMARGIN, 7
  RIGHTMARGIN, 168
  TOPMARGIN, 7
  BOTTOMMARGIN, 59
END

IDD_DIALOG3, DIALOG

```

```

BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 179
    TOPMARGIN, 7
    BOTTOMMARGIN, 125
END

```

```

IDD_DIALOG4, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 179
    TOPMARGIN, 7
    BOTTOMMARGIN, 114
END

```

```

IDD_PP10, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 180
    TOPMARGIN, 7
    BOTTOMMARGIN, 155
END

```

```

IDD_PP11, DIALOG
BEGIN
    LEFTMARGIN, 7
    RIGHTMARGIN, 237
    TOPMARGIN, 7
    BOTTOMMARGIN, 152
END

```

```

END
#endif // APSTUDIO_INVOKED

```

```

////////////////////////////////////
//
// Menu
//

```

```

IDR_MENU1 MENU DISCARDABLE
BEGIN

```

```

    POPUP "File"

```

```

    BEGIN

```

```

        MENUITEM "Open",

```

```

        ID_MENUITEM32771

```

```

        MENUITEM "Save",

```

```

        ID_MENUITEM32772

```

```

        MENUITEM "Export Param File",

```

```

        ID_FILE_EXPORTPARAMFILE

```

```

    END

```

```

END

```

```

////////////////////////////////////
//
// String Table
//

```

```

STRINGTABLE DISCARDABLE

```

```

BEGIN

```

```

    IDS_ABOUTBOX

```

```

    "&About KOJ's Amazing Town Crier..."

```

```

END

```

```

#endif // English (U.S.) resources

```

```

#ifndef APSTUDIO_INVOKED

```

```

////////////////////////////////////
//

```

```

// Generated from the TEXTINCLUDE 3 resource.
//

```

```

#define _AFX_NO_SPLITTER_RESOURCES

```

```

#define _AFX_NO_OLE_RESOURCES

```

0122

```

#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif
#include "res\sa.rc2" // non-Microsoft Visual C++ edited resources
#include "afxres.rc" // Standard components
#endif
////////////////////////////////////
#endif // not APSTUDIO_INVOKED

```

```

opt      cc,cex,cre,loc,so,mu,mex

DEFINE      FAST      '1'
DEFINE      M56362     '1'

TABLE_COUNTER EQU 10
RUN_COUNTER   EQU 10
INIT_COUNTER  EQU 10

org y(10):$100
org p(10):$980

; *****
; *****

SECTION      SUPER

; *****
; *****

-----
; HDCC detect and expand code
-----

include 'detect.asm'
include 'expgain.asm'

; *****
;
; Adopted from Motorola sample code.
;
; *****
;
; page      132,60
; include 'ioequ.asm'
; include 'vectors.asm'
; include 'mbiquad.asm'
; include 'mpeq.asm'
; include 'mshelv.asm'
; include 'mhipass.asm'
; include 'mlopass.asm'
; include 'allpass.asm'
; list

CONFIGURE_SPI MACRO

; move      #0,x0
; move      x0,X:M_HSAR                      ; clear I2C address

; M_HCKR
; bits 13:12 HFM1:0 noise filter setting (off)
; bit 1 CPOL
; bit 0 CPHA
;
; 00 xxxx xxxx xx01
; move      #1,x0
; move      #0,x0
; move      x0,X:M_HCKR
; commented out code should yeild a value of 0 for M_HCKR.

; M_HCSR
; bit 13:12 HCSR HRIE
; bit 11 HCSR HTIE
; bit 10 HCSR HBIE
; bit 9 HCSR Idle (no care)
; bit 8:7 HRQE Host-request enable (asserted if ready to receive)
; bit 6 HMST Master mode (disabled)
; bit 5 HFIFO FIFO control (disabled)
; bit 4 HCKFR Clock freeze (off)
; bit 3:2 HM1:0 24 bit mode
; bit 1 HI2C Enables I2C mode
; bit 0 HEN Enable port

```

0124

```

;
; xx01 0010 000x 1001
move    #$1209,x0
move    x0,X:M_HCSR

; set command to known value
move    #SPITXDEF,x0
move    x0,x:spixmit
; movep  x:spixmit,x:M_HTX

ENDM

```

CONFIGURE\_I2C MACRO

```

; turn off port
move    #0,x0
move    x0,X:M_HCSR

; leave address alone
; move    #0,x0
; move    x0,X:M_HSAR                ; clear I2C address

; commented out code should yeild a value of 0 for M_HCKR.
; M_HCKR
; bits 13:12 HFM1:0 noise filter setting on
;
; 11 xxxx xxxx xx00
move    #>$3000,x0
move    x0,X:M_HCKR

; M_HCSR
; bit 13:12 HCSR HRIE
; bit 11 HCSR HTIE
; bit 10 HCSR HBIE
; bit 9 HCSR Idle (no care)
; bit 8:7 HRQE Host-request enable (asserted if ready to receive)
; bit 6 HMST Master mode (disabled)
; bit 5 HFIFO FIFO control (disabled)
; bit 4 HCKFR Clock freeze (off)
; bit 3:2 HM1:0 24 bit mode
; bit 1 HI2C Enables I2C mode
; bit 0 HEN Enable port
;
; xx01 1100 101x 1010
move    #$1CAA,x0
move    x0,X:M_HCSR

; set command to known value
move    #0,x0
move    x0,x:spixmit
; movep  x:spixmit,x:M_HTX

ENDM

```

```

XDEF      fcontrol.
XREF      HDCD_DETECT_INIT,HDCD_DETECT,lsbcnt1
XREF      ltemp,rtemp
XREF      dcontrol
XREF      HDCD_GAIN_INIT,HDCD_DYNAMICS
XREF      detect
XREF      rmodel,bitcnt1

```

SPITXDEF equ \$444444

```

org      xi:
DRX_BUFF_BASE ds 2
ARX_BUFF_BASE ds 2

TX_BUFF_BASE ds 4      ; unprocessed followed by processed data

;RX_PTR ds 1
TX_PTR ds 1

```



```

flags      ds 1
scnt       ds 1
RightReceive equ 0
SPIReceive equ 1
SPIOverrun equ 2
SPIFIFOFull equ 3
SPIUnderrun equ 4
SPITransmit equ 5
SPIStateBit equ 6

```

```

spixmit     ds 1
spirecv     ds 1
shisaveA1   ds 1
shisaveX0   ds 1
shisaveR4   ds 1
shisaveN4   ds 1
NADA        ds 1
left        ds 1
right       ds 1
fcontrol    ds 1
indexData   ds 1
dpleft      ds 1
dpright     ds 1
stack       ds 20

```

```

; interleaved delay line
; independently located
org x(20):$200
DELAYLEN    EQU 128
sdelayleft  dsm DELAYLEN
sdelayright dsm DELAYLEN
; 2.9 mS delay line
; left
; right

```

```

org yi:
YNADA       ds 1
volume      ds 1
prefader    ds 1
bypass      ds 1
hdcdbp      ds 1
gainscalebp ds 1
ddxcompbp   ds 1
analogin    ds 1
avolume     ds 1
bypassmask  ds 1

BYPASS_NOTCH1 EQU 0
BYPASS_NOTCH2 EQU 1
BYPASS_NOTCH3 EQU 2
BYPASS_NOTCH4 EQU 3
BYPASS_HIPASS EQU 4
BYPASS_LOPASS EQU 5
BYPASS_LOSHELV EQU 6
BYPASS_HISHELV EQU 7
BYPASS_CONECRY EQU 8
BYPASS_ALLPASS EQU 9
BYPASS_DBNOTCH EQU 10
BYPASS_NLOPASS EQU 11

```

```

delayval    ds 1

```

```

; *****
; EQ/filter declarations
; *****
; smiley face shelf EQ
; left channel
DECLARE_LSH 1L,1.,-0.867
DECLARE_HSH 2L,1.,-0.867

; right channel
DECLARE_LSH 1R,1.,-0.867
DECLARE_HSH 2R,1.,-0.867

; resonance compensation PEQ

```

```

; left channel
DECLARE_PEQ 1L,1.,-0.9510565,0.7265425
DECLARE_PEQ 2L,1.,-0.91,0.5

DECLARE_PEQ 6L,1.,-0.9510565,0.7265425
DECLARE_PEQ 7L,1.,-0.91,0.5

; right channel
DECLARE_PEQ 1R,1.,-0.91,0.5
DECLARE_PEQ 2R,1.,-0.91,0.5

DECLARE_PEQ 6R,1.,-0.9510565,0.7265425
DECLARE_PEQ 7R,1.,-0.91,0.5

; standing wave rejection PEQ
; left channel
DECLARE_PEQ 3L,1.,-0.9999803,0.9937365
DECLARE_PEQ 4L,1.,-0.91,0.5
DECLARE_PEQ 5L,1.,-0.91,0.5

; right channel
DECLARE_PEQ 3R,1.,-0.9999803,0.9937365
DECLARE_PEQ 4R,1.,-0.91,0.5
DECLARE_PEQ 5R,1.,-0.91,0.5

; hi and lo pass filters
; left
DECLARE_HP hpleft,0.0099733,0.25,0.9987285 ; 70 Hz
DECLARE_LP lpleft,1.1921856,0.6303886,1.4112871 ; 14000 Hz / q = 0.22

; right
DECLARE_HP hpright,0.0099733,0.25,0.9987285 ; 70 Hz
DECLARE_LP lpright,1.1921856,0.6303886,1.4112871 ; 14000 Hz

; new lopass
; left
DECLARE_LP lpleft2,1.,0.,0.
; right
DECLARE_LP lpright2,1.,0.,0.

; allpass filters
DECLARE_AP apleft,-0.91,0.5
DECLARE_AP apright,-0.91,0.5

; Keith new notch with 2 gain around it.
DECLARE_PEQ NT1L,1.,-0.9999803,0.9937365
DECLARE_PEQ NT2L,1.,-0.91,0.5
DECLARE_PEQ NT3L,1.,-0.91,0.5

DECLARE_PEQ NT1R,1.,-0.9999803,0.9937365
DECLARE_PEQ NT2R,1.,-0.91,0.5
DECLARE_PEQ NT3R,1.,-0.91,0.5

```

```

;*****
START      org      p:$100
main
    ori      #03,mr                ; mask interrupts
    movep    #05000B,X:M_PCTL      ;
    move     #0,omr
    movec    #0,sp
    movep    #000003,X:M_IPRP      ; reset hardware stack pointer
    movep    #000007,X:M_IPRP      ; ESAI int's enabled and top Priority
    move     #stack,r6             ; SHI (1) and ESAI (2) int's enabled
    move     #-1,m6                ; initialize stack pointer
                                   ; linear addressing

    move     #0,x0
    move     x0,x:DRX_BUFF_BASE
    move     x0,x:DRX_BUFF_BASE+1
    move     x0,x:ARX_BUFF_BASE

```

```

move    x0,x:TX_BUFF_BASE+1
move    x0,x:TX_BUFF_BASE
move    x0,x:TX_BUFF_BASE+1
move    x0,x:TX_BUFF_BASE+2
move    x0,x:TX_BUFF_BASE+3

```

```

clr     a
move    a,x:flags
move    a,x:scnt

```

```

; HDCD decoder initialize
move    #>1,x0
move    x0,x:fcontrol
move    #>16,x0
move    x0,x:lsbcnt1
jsr     HDCD_DETECT_INIT
move    #>1,x0
move    x0,x:dcontrol
jsr     HDCD_GAIN_INIT

```

```

move    #-1,m0
move    #-1,m1
move    #-1,m2
move    #-1,m3
move    #-1,m4
move    #-1,m5
move    #-1,m6
move    #-1,m7

```

```

; set up volume and bypass switches
; output volume is set to zero so that filter parameters can
; be downloaded, followed by volume being set

```

```

move    #>$800000,x0
move    x0,y:volume
move    x0,y:prefader
move    x0,y:avolume
move    #>$7FFFFFFF,x0
move    x0,y:bypass
move    #>$0,x0
move    x0,y:hdcdbp
move    x0,y:gainscalebp
move    x0,y:ddxcompbp
move    x0,y:analogin
move    x0,y:bypassmask
move    x0,y:delayval
move    x0,x:indexData

```

```

move    #>sdelayleft,x0
move    x0,x:dpleft
move    #>sdelayright,x0
move    x0,x:dpright

```

```

; initialize EQ params
INIT_PEQ_PARAM 1L
INIT_PEQ_PARAM 1R
INIT_PEQ_PARAM 2L
INIT_PEQ_PARAM 2R
INIT_PEQ_PARAM 3L
INIT_PEQ_PARAM 3R
INIT_PEQ_PARAM 4L
INIT_PEQ_PARAM 4R
INIT_PEQ_PARAM 5L
INIT_PEQ_PARAM 5R
INIT_PEQ_PARAM 6L
INIT_PEQ_PARAM 6R
INIT_PEQ_PARAM 7L
INIT_PEQ_PARAM 7R

```

```

INIT_PEQ_PARAM NT1L
INIT_PEQ_PARAM NT2L
INIT_PEQ_PARAM NT3L

```

```

INIT_PEQ_PARAM NT1R
INIT_PEQ_PARAM NT2R
INIT_PEQ_PARAM NT3R

```

```

INIT_SH_PARAM 1L
INIT_SH_PARAM 1R
INIT_SH_PARAM 2L
INIT_SH_PARAM 2R

```

```

INIT_LP_PARAM lpleft
INIT_LP_PARAM lpright

```

```

INIT_LP2_PARAM lpleft2
INIT_LP2_PARAM lpright2

```

```

INIT_AP_PARAM apleft
INIT_AP_PARAM apright

```

```

; hi-pass does not need initializing as all params are downloaded

```

```

; setup serial host interface
;; CONFIGURE_SPI
CONFIGURE_I2C

```

```

-----
; FST/FSR and SCKT/SCKR are generated from the PLD
; and fed to the DSP, A/D and D/A converters
-----

```

```

;put esai in reset state.

```

```

; IF @DEF('ANALOGIN')==0
movep #0000000,x:M_PCRS ; MLS 12/20/97
movep #0000000,x:M_PRRS ; MLS 12/20/97
; ENDIF

```

```

movep #0c0200,x:M_TCCR
;FST is input (bit22=0)
;external clock source drives SCKT (bit21=0)
;negative FST polarity (bit19=1)
;data & FST clocked out on rising edge (bit18=1)
;2 words per frame (bit13:9=00001)

```

```

movep #0c0200,x:M_RCCR
;FSR is input (bit22=0)
;external clock source drives SCKR (bit21=0)
;negative FSR polarity (bit19=1)
;data & FSR clocked in on rising edge (bit18=0)BAK(121997)
;2 words per frame (bit13:9=00001)

```

```

movep #0000000,x:M_SAICR

```

```

movep    #$d1700,x:M_RCR
;RX1, RX0 enabled                      (bit1:0=11)
;RX2, RX3 disabled                     (bit3:2=00)
;reserved                              (bit5:4=00)
;MSB shifted first                     (bit6=0)
;word left-aligned                     (bit7=0)
;network mode                          (bit9:8=01)
;32-bit slot length, 24-bit word length (bit14:10=11111)
;word-length frame sync                (bit15=0)
;frame sync occurs 1 clock cycle earlier (bit16=1)
;reserved                              (bit19:17=000)
;RLIE, RIE, REIE enabled               (bit23:20=0101)
;bit23 RLIE
;bit22 RIE
;bit21 REDIE
;bit20 REIE

```

```

movep    #$d13d00,x:M_TCR
;TX0, TX1 enabled                      ;MLS 12/20/97
;TX2, TX3, TX4, TX5 disabled          (bit3:0=0011)
;MSB shifted first                     (bit5:4=00)
;word left-aligned                     (bit6=0)
;network mode                          (bit7=0)
;32-bit slot length, 24-bit word length (bit9:8=01)
;word length frame sync                (bit14:10=11111)
;frame sync occurs 1 clock cycle earlier (bit15=0)
;reserved                              (bit16=0)
;TLIE, TIE, TEIE enabled               (bit19:17=000)
;bit23 TLIE                           (bit23:20=0101)
;bit22 TIE
;bit21 TEDIE
;bit20 TEIE

```

```

movep    #$000edb,x:M_PCRC              ; MLS 12/20/97
movep    #$000edb,x:M_PRRC              ; MLS 12/20/97

```

```

movep    #$ffffff,x:M_RSMA              ;MLS 12/20/97
movep    #$ffffff,x:M_RSMB              ;MLS 12/20/97

```

```

movep    #$000003,x:M_TSMA
movep    #$000003,x:M_TSMB
movep    #$000000,x:M_TX0                ;zero out transmitter 0
movep    #$000000,x:M_TX1                ;zero out transmitter 1
movep    #$000000,x:M_TX2                ;zero out transmitter 2
movep    #$000000,x:M_TX3                ;zero out transmitter 3
bset     #0,x:M_TCR                      ;now enable TX0
bset     #1,x:M_TCR                      ;now enable TX1
bset     #2,x:M_TCR                      ;now enable TX2
bset     #3,x:M_TCR                      ;now enable TX3
bset     #2,x:M_RCR                      ;RE2 and TE3 must be set to enable
;TX3 and disable RX2

```

```

; turn on serial host interface
bset     #0,x:M_HCSR

```

```

andi     #$FC,mr                        ;enable all interrupt levels
;clear scaling bits

```

```

bclr     #RightReceive,X:flags

```

```

;
jclr     #RightReceive,X:flags,*
;
bclr     #RightReceive,X:flags
;
jclr     #RightReceive,X:flags,*
;
move     #>RX_BUFF_BASE,x0
;
move     x0,x:RX_PTR
;
move     #>TX_BUFF_BASE,x0
;
move     x0,x:TX_PTR

```

```

;-----
; Main loop
;-----

```

```

LOOP

```

0130

```

jclr    #RightReceive,x:flags,*
bclr    #RightReceive,X:flags

btst     #22,y:analogin
.if      <CC>
    move    x:DRX_BUFF_BASE,x0        ;receive left
    move    x:DRX_BUFF_BASE+1,x1      ;receive right
.else
    move    x:ARX_BUFF_BASE,x0        ;receive left
    move    x:ARX_BUFF_BASE+1,x1      ;receive right
.ENDI

; store unprocessed (with volume) output first
move     y:avolume,y0
mpyr     y0,x0,a
mpyr     y0,x1,b
; MM 5/20/99 Swapped channels
tfr      x0,a      a,x:TX_BUFF_BASE+2 ;transmit left
tfr      x1,b      b,x:TX_BUFF_BASE   ;transmit right

; write input into delay line
move     #DELAYLEN-1,m0
move     #DELAYLEN-1,m1
move     x:dpleft,r0
move     x:dpright,r1
move     y:delayval,y0
move     #>DELAYLEN,x0
mpy      x0,y0,a      a,x:(r0)-
move     b,x:(r1)-
move     a,n0
move     a,n1
move     x:(r0+n0),a
move     x:(r1+n1),b
move     #-1,m0
move     #-1,m1
move     r0,x:dpleft
move     r1,x:dpright

JSR      STEREO_PROCESS

; MM 5/20/99 Swapped channels
move     b,x:TX_BUFF_BASE+1          ;transmit left
move     a,x:TX_BUFF_BASE+3          ;transmit right

move     #>1,x0
move     x:scnt,a
add      x0,a
move     a1,x:scnt                    ; don't saturate

jmp      LOOP

```

```

;-----
; Subroutines
;-----

```

```

STEREO_PROCESS

```

```

move     a,x:left
move     b,x:right

```

```

; copy parameters from left channel EQs to right channel EQs

```

```

COPY_SH_PARAM 1L,1R
COPY_SH_PARAM 2L,2R

```

```

COPY_PEQ_PARAM 1L,1R
COPY_PEQ_PARAM 2L,2R
COPY_PEQ_PARAM 3L,3R
COPY_PEQ_PARAM 4L,4R
COPY_PEQ_PARAM 5L,5R
COPY_PEQ_PARAM 6L,6R
COPY_PEQ_PARAM 7L,7R

```

0131

```

COPY_PEQ_PARAM NT1L,NT1R
COPY_PEQ_PARAM NT2L,NT2R
COPY_PEQ_PARAM NT3L,NT3R

```

```

COPY_HP_PARAM hpleft,hpright
COPY_LP_PARAM lpleft,lpright
COPY_LP_PARAM lpleft2,lpright2

```

```

COPY_AP_PARAM apleft,apright

```

```

; HD CD processing
;; jset #22,y:hdcdbp,doeq

```

```

move x:left,a
move x:right,b
move a,x:ltemp
move b,x:rtemp

```

```

clr b
move #>1,x0
move y:gainscalebp,a
tst a
teq x0,b
move b1,x:dcontrol

```

```

jsr HD CD_DETECT

```

```

; destroy code
move #>3,x0
move x:rmodel,a
cmp x0,a
bne skip_code_dest
move #>8,x0
move x:bitcnt1,a
cmp x0,a
bne skip_code_dest

```

```

move x:ltemp,a
eor #>$000100,a
move a,x:ltemp
move a,x:left

```

```

skip_code_dest:

```

```

move #-1,m0
move #-1,m4
move #4,n4
jsr HD CD_DYNAMICS

```

```

move x:ltemp,a
move x:rtemp,b
btst #22,y:hdcdbp
.if <CC>
move a,x:left
move b,x:right
move x:detect,x0
move x0,x:spixmit

```

```

.ELSE
move #0,x0
move x0,x:spixmit
.ENDI

```

```

; jmp doeq1

```

```

;doeq:

```

```

;
; move #0,x0
; move x0,x:spixmit

```

```

doeq1:

```

```

jset #22,y:bypass,dopostvol

```

```

move x:left,x1

```

0132

```

move      x:right,y1
move      y:prefader,x0
mpyr      -x0,x1,a
mpyr      -x0,y1,b

tfr      a,b      b,x:right

; smiley face
jset      #BYPASS_LOSHELV,y:bypassmask,_skipls1
LSH 1L
_skipls1
jset      #BYPASS_HISHELV,y:bypassmask,_skiphs1
HSH 2L
_skiphs1
; cut-off response
jset      #BYPASS_HIPASS,y:bypassmask,_skiphpl
HP hpleft
_skiphpl
jset      #BYPASS_LOPASS,y:bypassmask,_skiplpl
LP lpleft
_skiplpl
jset      #BYPASS_NLOPASS,y:bypassmask,_skiplp21
LP lpleft2
_skiplp21
; resonance EQ
jset      #BYPASS_NOTCH1,y:bypassmask,_skip11
PEQ 1L
_skip11
jset      #BYPASS_NOTCH2,y:bypassmask,_skip21
PEQ 2L
_skip21
jset      #BYPASS_NOTCH3,y:bypassmask,_skip31
PEQ 6L
_skip31
jset      #BYPASS_NOTCH4,y:bypassmask,_skip41
PEQ 7L
_skip41
; cone-cry
jset      #BYPASS_CONECRY,y:bypassmask,_skipccl
PEQ 5L
PEQ 3L
PEQ 4L
_skipccl
; double-tuned notch
jset      #BYPASS_DBNOTCH,y:bypassmask,_skipdbnl
PEQ NT1L
PEQ NT2L
PEQ NT3L
_skipdbnl
; all-pass
jset      #BYPASS_ALLPASS,y:bypassmask,_skipapl
AP apleft
_skipapl

move      b,x:left
move      x:right,b

; smiley face
jset      #BYPASS_LOSHELV,y:bypassmask,_skiplsr
LSH 1R
_skiplsr
jset      #BYPASS_HISHELV,y:bypassmask,_skiphsr
HSH 2R
_skiphsr
; cut-off response
jset      #BYPASS_HIPASS,y:bypassmask,_skiphpr
HP hpright
_skiphpr
jset      #BYPASS_LOPASS,y:bypassmask,_skiplpr
LP lpright
_skiplpr
jset      #BYPASS_NLOPASS,y:bypassmask,_skiplp2r

```



```

        LP lpright2
_skiplp2r
        ; resonance EQ
        jset    #BYPASS_NOTCH1,y:bypassmask,_skip1r
        PEQ 1R
_skip1r
        jset    #BYPASS_NOTCH2,y:bypassmask,_skip2r
        PEQ 2R
_skip2r
        jset    #BYPASS_NOTCH3,y:bypassmask,_skip3r
        PEQ 6R
_skip3r
        jset    #BYPASS_NOTCH4,y:bypassmask,_skip4r
        PEQ 7R
_skip4r
        ; cone-cry
        jset    #BYPASS_CONECRY,y:bypassmask,_skipccr
        PEQ 5R
        PEQ 3R
        PEQ 4R
_skipccr
        ; double-tuned notch
        jset    #BYPASS_DBNOTCH,y:bypassmask,_skipdbnr
        PEQ NT1R
        PEQ NT2R
        PEQ NT3R
_skipdbnr
        ; all-pass
        jset    #BYPASS_ALLPASS,y:bypassmask,_skipapr
        AP apright
_skipapr
        move    b,x:right

```

depostvol:

plainvol:

```

move    y:volume,y0
move    x:left,x0
move    x:right,x1
mpyr    -x0,y0,a
mpyr    -x1,y0,b
rts

```



1. *Phragmites australis* (Cav.) Trin. ex Steud.  
 2. *Spartina patens* (Muhl.) Bosc.  
 3. *Scirpus americanus* (L.) Link.  
 4. *Distichlis spicata* (L.) Nees  
 5. *Eleocharis acicularis* (L.) Rostk Schmidt  
 6. *Eleocharis obtusa* (L.) Nees  
 7. *Eleocharis tenuis* (L.) Rostk Schmidt  
 8. *Eleocharis palustris* (L.) Rostk Schmidt  
 9. *Eleocharis acicularis* (L.) Rostk Schmidt  
 10. *Eleocharis obtusa* (L.) Nees  
 11. *Eleocharis tenuis* (L.) Rostk Schmidt  
 12. *Eleocharis palustris* (L.) Rostk Schmidt  
 13. *Eleocharis acicularis* (L.) Rostk Schmidt  
 14. *Eleocharis obtusa* (L.) Nees  
 15. *Eleocharis tenuis* (L.) Rostk Schmidt  
 16. *Eleocharis palustris* (L.) Rostk Schmidt  
 17. *Eleocharis acicularis* (L.) Rostk Schmidt  
 18. *Eleocharis obtusa* (L.) Nees  
 19. *Eleocharis tenuis* (L.) Rostk Schmidt  
 20. *Eleocharis palustris* (L.) Rostk Schmidt  
 21. *Eleocharis acicularis* (L.) Rostk Schmidt  
 22. *Eleocharis obtusa* (L.) Nees  
 23. *Eleocharis tenuis* (L.) Rostk Schmidt  
 24. *Eleocharis palustris* (L.) Rostk Schmidt  
 25. *Eleocharis acicularis* (L.) Rostk Schmidt  
 26. *Eleocharis obtusa* (L.) Nees  
 27. *Eleocharis tenuis* (L.) Rostk Schmidt  
 28. *Eleocharis palustris* (L.) Rostk Schmidt  
 29. *Eleocharis acicularis* (L.) Rostk Schmidt  
 30. *Eleocharis obtusa* (L.) Nees  
 31. *Eleocharis tenuis* (L.) Rostk Schmidt  
 32. *Eleocharis palustris* (L.) Rostk Schmidt  
 33. *Eleocharis acicularis* (L.) Rostk Schmidt  
 34. *Eleocharis obtusa* (L.) Nees  
 35. *Eleocharis tenuis* (L.) Rostk Schmidt  
 36. *Eleocharis palustris* (L.) Rostk Schmidt  
 37. *Eleocharis acicularis* (L.) Rostk Schmidt  
 38. *Eleocharis obtusa* (L.) Nees  
 39. *Eleocharis tenuis* (L.) Rostk Schmidt  
 40. *Eleocharis palustris* (L.) Rostk Schmidt  
 41. *Eleocharis acicularis* (L.) Rostk Schmidt  
 42. *Eleocharis obtusa* (L.) Nees  
 43. *Eleocharis tenuis* (L.) Rostk Schmidt  
 44. *Eleocharis palustris* (L.) Rostk Schmidt  
 45. *Eleocharis acicularis* (L.) Rostk Schmidt  
 46. *Eleocharis obtusa* (L.) Nees  
 47. *Eleocharis tenuis* (L.) Rostk Schmidt  
 48. *Eleocharis palustris* (L.) Rostk Schmidt  
 49. *Eleocharis acicularis* (L.) Rostk Schmidt  
 50. *Eleocharis obtusa* (L.) Nees  
 51. *Eleocharis tenuis* (L.) Rostk Schmidt  
 52. *Eleocharis palustris* (L.) Rostk Schmidt  
 53. *Eleocharis acicularis* (L.) Rostk Schmidt  
 54. *Eleocharis obtusa* (L.) Nees  
 55. *Eleocharis tenuis* (L.) Rostk Schmidt  
 56. *Eleocharis palustris* (L.) Rostk Schmidt  
 57. *Eleocharis acicularis* (L.) Rostk Schmidt  
 58. *Eleocharis obtusa* (L.) Nees  
 59. *Eleocharis tenuis* (L.) Rostk Schmidt  
 60. *Eleocharis palustris* (L.) Rostk Schmidt  
 61. *Eleocharis acicularis* (L.) Rostk Schmidt  
 62. *Eleocharis obtusa* (L.) Nees  
 63. *Eleocharis tenuis* (L.) Rostk Schmidt  
 64. *Eleocharis palustris* (L.) Rostk Schmidt  
 65. *Eleocharis acicularis* (L.) Rostk Schmidt  
 66. *Eleocharis obtusa* (L.) Nees  
 67. *Eleocharis tenuis* (L.) Rostk Schmidt  
 68. *Eleocharis palustris* (L.) Rostk Schmidt  
 69. *Eleocharis acicularis* (L.) Rostk Schmidt  
 70. *Eleocharis obtusa* (L.) Nees  
 71. *Eleocharis tenuis* (L.) Rostk Schmidt  
 72. *Eleocharis palustris* (L.) Rostk Schmidt  
 73. *Eleocharis acicularis* (L.) Rostk Schmidt  
 74. *Eleocharis obtusa* (L.) Nees  
 75. *Eleocharis tenuis* (L.) Rostk Schmidt  
 76. *Eleocharis palustris* (L.) Rostk Schmidt  
 77. *Eleocharis acicularis* (L.) Rostk Schmidt  
 78. *Eleocharis obtusa* (L.) Nees  
 79. *Eleocharis tenuis* (L.) Rostk Schmidt  
 80. *Eleocharis palustris* (L.) Rostk Schmidt  
 81. *Eleocharis acicularis* (L.) Rostk Schmidt  
 82. *Eleocharis obtusa* (L.) Nees  
 83. *Eleocharis tenuis* (L.) Rostk Schmidt  
 84. *Eleocharis palustris* (L.) Rostk Schmidt  
 85. *Eleocharis acicularis* (L.) Rostk Schmidt  
 86. *Eleocharis obtusa* (L.) Nees  
 87. *Eleocharis tenuis* (L.) Rostk Schmidt  
 88. *Eleocharis palustris* (L.) Rostk Schmidt  
 89. *Eleocharis acicularis* (L.) Rostk Schmidt  
 90. *Eleocharis obtusa* (L.) Nees  
 91. *Eleocharis tenuis* (L.) Rostk Schmidt  
 92. *Eleocharis palustris* (L.) Rostk Schmidt  
 93. *Eleocharis acicularis* (L.) Rostk Schmidt  
 94. *Eleocharis obtusa* (L.) Nees  
 95. *Eleocharis tenuis* (L.) Rostk Schmidt  
 96. *Eleocharis palustris* (L.) Rostk Schmidt  
 97. *Eleocharis acicularis* (L.) Rostk Schmidt  
 98. *Eleocharis obtusa* (L.) Nees  
 99. *Eleocharis tenuis* (L.) Rostk Schmidt  
 100. *Eleocharis palustris* (L.) Rostk Schmidt

```

;-----
; SHI interrupts
;-----

```

```

; ; ---- SHI/I2C receive ISR
shi_rx_isr

```

```

move    a1,x:shisaveA1
move    x0,x:shisaveX0
move    r4,x:shisaveR4
move    n4,x:shisaveN4
movep   x:M_HRX,a1
jset    #SPIStateBit,x:flags,_gotData

```

```
; got index
```

```

move    a1,x:indexData
bset    #SPIStateBit,x:flags
jmp     _exitInt

```

```
; got data
```

```
_gotData:
```

```
move    x:indexData,n4
move    #pptrs,r4
```

0135

```

        bclr      #SPIStateBit,x:flags
        movem     p:(r4+n4),r4
        move      a1,y:(r4)

_exitInt:
        move      x:shisaveA1,a1
        move      x:shisaveX0,x0
        move      x:shisaveR4,r4
        move      x:shisaveN4,n4
        rti

; ---- SHI/I2C receive overrun error
shi_rxe_isr
        movep     x:M_HCSR,x:NADA
        movep     x:M_HRX,x:NADA
        bset      #SPIOverrun,x:flags
        bclr      #SPIStateBit,x:flags
        rti

; SHI Receive FIFO Full
shi_rxf_isr
        movep     x:M_HCSR,x:NADA
        movep     x:M_HRX,x:NADA
        bset      #SPIFIFOFull,x:flags
        bclr      #SPIStateBit,x:flags
        rti

; SHI Transmit Data
shi_txu_isr ;SHI Transmit Underrun Error
        bset      #SPIUnderrun,x:flags
        movep     x:M_HCSR,x:NADA
        movep     x:spixmit,x:M_HTX
        bclr      #SPIStateBit,x:flags
        rti

shi_tx_isr:
        movep     x:spixmit,x:M_HTX
        bset      #SPITransmit,x:flags
        rti

;SHI Bus Error
shi_bus_error:
;        movep     x:M_HCSR,x:NADA
;        bset      #SPIReceive+4,x:flags
;        ; reset shi
        bclr      #0,x:M_HCSR
        nop
        nop
        bset      #0,x:M_HCSR
        bclr      #SPIStateBit,x:flags
        nop
        rti

;-----
; Subroutines
;-----
;
;        include 'isr_dig.asm'
;-----
; Interrupt Service Routines
;-----

esai_txe_isr ; ESAI TRANSMIT ISR
        bclr      #14,x:M_SAISR ; Read SAISR to clear transmit
esai_tx_isr ; underrun error flag

        jset      #13,x:M_SAISR,TxLeftSlot

        movep     x:TX_BUFF_BASE+2,x:M_TX1 ; write unprocessed data
        movep     x:TX_BUFF_BASE+3,x:M_TX0
        movep     x:TX_BUFF_BASE+3,x:M_TX2

```

```

        movep    x:TX_BUFF_BASE+3,x:M_TX3
        rti

TxLeftSlot

        movep    x:TX_BUFF_BASE,x:M_TX1
        movep    x:TX_BUFF_BASE+1,x:M_TX0      ; write unprocessed data
        movep    x:TX_BUFF_BASE+1,x:M_TX2
        movep    x:TX_BUFF_BASE+1,x:M_TX3
        rti

esai_txls_isr
;         move    r0,x:(r6)+      ; ESAI TRANSMIT LAST SLOT ISR
;         move    #TX_BUFF_BASE,r0 ; Save r0 to the stack
;         move    r0,x:TX_PTR     ; Reset pointer
;                                     ; Reset tx buffer pointer just in
;                                     ; case it was corrupted
;         move    x:-(r6),r0      ; Restore r0
        rti

esai_rxe_isr
        bclr     #7,x:M_SAISR      ; ESAI RECEIVE ISR
        overrun error flag        ; Read SAISR to clear receive

esai_rx_isr
;                                     ; overrun error flag

        jset     #$6,x:M_SAISR,LeftSlot
        bset     #RightReceive,x:flags ; if right channel data then set flag
        movep    x:M_RX0,x:ARX_BUFF_BASE+1
        movep    x:M_RX1,x:DRX_BUFF_BASE+1
        rti

LeftSlot

        movep    x:M_RX0,x:ARX_BUFF_BASE
        movep    x:M_RX1,x:DRX_BUFF_BASE
        rti

esai_rxls_isr
;         move    r0,x:(r6)+      ; ESAI RECEIVE LAST SLOT ISR
;         move    #RX_BUFF_BASE,r0 ; Save r0 to the stack
;                                     ; Reset rx buffer pointer just in
;                                     ; case it was corrupted
;         move    r0,x:RX_PTR     ; Update rx buffer pointer
;         move    x:-(r6),r0      ; Restore r0
        rti

; variable look up table

pptrs
dc volume
dc sh_gamma_1L
dc sh_k_1L
dc sh_gamma_2L
dc sh_k_2L
dc peq_gamma_1L
dc peq_beta_1L
dc peq_k_1L
dc peq_gamma_2L
dc peq_beta_2L
dc peq_k_2L
dc peq_gamma_3L
dc peq_beta_3L
dc peq_k_3L
dc peq_gamma_4L
dc peq_beta_4L
dc peq_k_4L
dc peq_gamma_5L
dc peq_beta_5L
dc peq_k_5L
dc prefader
dc bypass
dc hp_scale_hpleft
dc hp_fc_hpleft
dc hp_qc_hpleft

```

0137

dc lp\_scale\_lpleft  
dc lp\_a2\_lpleft  
dc lp\_a1\_lpleft  
dc hdcdbp  
dc gainscalebp  
dc ddxcompbp  
dc peq\_gamma\_6L  
dc peq\_beta\_6L  
dc peq\_k\_6L  
dc peq\_gamma\_7L  
dc peq\_beta\_7L  
dc peq\_k\_7L  
dc analogin  
dc avolume  
dc bypassmask  
dc delayval  
dc ap\_gamma\_aleft  
dc ap\_beta\_aleft  
dc lp\_scale\_lpleft2  
dc lp\_a2\_lpleft2  
dc lp\_a1\_lpleft2  
dc lp\_b2\_lpleft2  
dc lp\_b1\_lpleft2  
dc peq\_gamma\_NT1L  
dc peq\_beta\_NT1L  
dc peq\_k\_NT1L  
dc peq\_gamma\_NT2L  
dc peq\_beta\_NT2L  
dc peq\_k\_NT2L  
dc peq\_gamma\_NT3L  
dc peq\_beta\_NT3L  
dc peq\_k\_NT3L  
dc YNADA  
dc YNADA  
dc YNADA

ENDSEC

end

```
SH_SCALE_FACTOR EQU 3
SH_SCALE_DIVIDE EQU (1<<SH_SCALE_FACTOR)
```

```
;
; k is gain
; k = 0      => lo/hi pass filter
; 0 < k < 1  => cut
; k = 1      => pass-thru
; k > 1      => boost
;
```

```
; gamma drives critical frequency
;
```

```
DECLARE_LSH MACRO      name,k,gamma
```

```
    org xi:
sh_x\_name      dc 0      ; x(n-1)
sh_y\_name      dc 0      ; y(n-1)
```

```
    org yi:
sh_gamma\_name   dc gamma
sh_minus_one\_name dc -1.0
sh_k\_name       dc k/SH_SCALE_DIVIDE
sh_i\_name       dc 1./SH_SCALE_DIVIDE
```

```
ENDM
```

```
DECLARE_HSH MACRO      name,k,gamma
```

```
    DECLARE_LSH name,k,gamma
```

```
ENDM
```

```
COPY_SH_PARAM MACRO      from,to
```

```
    move    y:sh_gamma\_from,x0
    move    x0,y:sh_gamma\_to
    move    y:sh_k\_from,x0
    move    x0,y:sh_k\_to
```

```
ENDM
```

```
INIT_SH_PARAM MACRO      name
```

```
    move    #>(-1.0),x0
    move    x0,y:sh_minus_one\_name
    move    #>(1./SH_SCALE_DIVIDE),x0
    move    x0,y:sh_k\_name
    move    x0,y:sh_i\_name
```

```
ENDM
```

```
;
; input data is in x0
; input is scaled by 0.5 to prevent internal clipping in the all-pass
;
; computes all-pass:
;
; y(n) = - gamma * x(n) - x(n-1) - gamma * y(n-1) (lo-shelv)
; y(n) = gamma * x(n) + x(n-1) - gamma * y(n-1) (hi-shelv)
;
; all-pass output is scaled by gain:
;
; output = ((1+k)/2) * x(n) + ((1-k)/2) * y(n)
;
; assumes:
; m0,m4,m5 = -1
;
```

0139

LSH MACRO            name

```

asr      b                      #sh_gamma_\name,r4
rnd      b                      #sh_x_\name,r0
move     b,x0                   y:(r4)+,y0
; compute y(n)
mpy      -x0,y0,b               x:(r0)+,x1       y:(r4)+,y1
mac      x1,y1,b               x:(r0)-,x1
macr     -y0,x1,b              x0,x:(r0)+       y:(r4)+,y0
; all-pass output in b, now scale by gain/cut factor
mpy      x0,y0,b               b,x:(r0)-       b,y1
mac      -y1,y0,b              y:(r4)+,y0
mac      x0,y0,b
mac      y1,y0,b
; now scale output to get real result
asl      #(SH_SCALE_FACTOR),b,b
rnd      b

```

ENDM

HSH MACRO            name

```

asr      b                      #sh_gamma_\name,r4
rnd      b                      #sh_x_\name,r0
move     b,x0                   y:(r4)+,y0
; compute y(n)
mpy      x0,y0,b               x:(r0)+,x1       y:(r4)+,y1
mac      -x1,y1,b              x:(r0)-,x1
macr     -y0,x1,b              x0,x:(r0)+       y:(r4)+,y0
; all-pass output in b, now scale by gain/cut factor
mpy      x0,y0,b               b,x:(r0)-       b,y1
mac      -y1,y0,b              y:(r4)+,y0
mac      x0,y0,b
mac      y1,y0,b
; now scale output to get real result
asl      #(SH_SCALE_FACTOR),b,b
rnd      b

```

ENDM

```

PEQ_SCALE_FACTOR    EQU 3
PEQ_SCALE_DIVIDE    EQU (1<<PEQ_SCALE_FACTOR)

```

```

;
; k is gain
;   k = 0      => notch filter
;   0 < k < 1  => cut
;   k = 1      => pass-thru
;   k > 1      => boost
;
; gamma drives critical frequency
;
; beta defines Q
;

```

```

DECLARE_PEQ MACRO      name,k,gamma,beta

```

```

    org xi:
peq_y_\name          dc 0
                      dc 0
peq_x_\name          dc 0
                      dc 0

    org yi:
peq_gamma_\name      dc gamma
peq_beta_\name       dc beta
peq_k_\name          dc k/PEQ_SCALE_DIVIDE
peq_l_\name          dc 1./PEQ_SCALE_DIVIDE
peq_v_\name          dc 0
peq_w_\name          dc 0

```

```

ENDM

```

```

COPY_PEQ_PARAM MACRO      from,to

```

```

    move    y:peq_gamma_\from,x0
    move    x0,y:peq_gamma_\to
    move    y:peq_beta_\from,x0
    move    x0,y:peq_beta_\to
    move    y:peq_k_\from,x0
    move    x0,y:peq_k_\to

```

```

ENDM

```

```

INIT_PEQ_PARAM MACRO      name

```

```

    move    #>(1./PEQ_SCALE_DIVIDE),x0
    move    x0,y:peq_k_\name
    move    x0,y:peq_l_\name

```

```

ENDM

```

```

;
; input data is in b
; input is scaled by 0.5 to prevent internal clipping in the all-pass
;
; computes all-pass:
;
; v(n) = gamma * y(n-1) + y(n-2) - gamma * v(n-1)
; w(n) = gamma * x(n-1) + x(n-2) - gamma * w(n-1)
; y(n) = w(n) + beta * x(n) - beta * v(n)
;
; all-pass output is scaled by gain:
;
; output = ((1+k)/2) * x(n) + ((1-k)/2) * y(n)
;
; assumes:
;   m0,m4,m5 = -1
;

```

0141



PEQ MACRO

name

```

asr      b      #peq_y_\name,r0
move     #peq_gamma_\name,r4
move     #peq_v_\name,r5
move     b,x0
move     x:(r0)+,a      y:(r4)+,y0
move     x:(r0)+,x1      y:(r5)+,y1
; compute v(n)
mac      x1,y0,a      x:(r0)+,b
mac      -y1,y0,a      x:(r0),x1      y:(r5)-,y1
; compute w(n), v(n) is now in a
mac      x1,y0,b      a,x1      a,y:(r5)+
mac      -y1,y0,b      x:(r0),a      y:(r4)+,y0
; compute y(n), w(n) is now in b
mac      -x1,y0,b      x0,x:(r0)-      b,y:(r5)-
mac      x0,y0,b      a,x:(r0)-      y:(r4)+,y0
; all-pass output in b, now scale by gain/cut factor
mpy      x0,y0,b      x:(r0)-,a      b,y1
mac      -y1,y0,b      y:(r4)+,y0
mac      x0,y0,b      a,x:(r0)+
mac      y1,y0,b      y1,x:(r0)-
; now scale output to get real result
asl      #(PEQ_SCALE_FACTOR),b,b
rnd      b

```

ENDM

0142

```

LP_SCALE_FACTOR EQU 6
LP_SCALE_DIVIDE EQU (1<<LP_SCALE_FACTOR)

```

```

;
;
;   modified chamberlin lo-pass (both x(n) and x(n-1))
;
;   q also defines gain
;   input is scaled to avoid internal clipping
;
;   f = 2*sin(w/2)  f only valid when less than 1.
;
;   0 < q < 1
;
;   implemented as a direct form biquad with coefficients and data scaled.
;   needs headroom for gain
;

```

```

DECLARE_LP MACRO      name,ca1,ca2,cb

```

```

    org xi:
lp_w_\name          ds 2

    org yi:
lp_scale_\name      dc cb/16.
lp_a2_\name         dc ca2/2.
lp_a1_\name         dc ca1/2.
lp_b2_\name         dc 0.0
lp_b1_\name         dc 1.0/2.

```

```

    ENDM

```

```

COPY_LP_PARAM MACRO      from,to

```

```

    move    y:lp_scale_\from,x0
    move    x0,y:lp_scale_\to
    move    y:lp_a2_\from,x0
    move    x0,y:lp_a2_\to
    move    y:lp_a1_\from,x0
    move    x0,y:lp_a1_\to
    move    y:lp_b2_\from,x0
    move    x0,y:lp_b2_\to
    move    y:lp_b1_\from,x0
    move    x0,y:lp_b1_\to

```

```

    ENDM

```

```

INIT_LP_PARAM MACRO      name

```

```

    move    #0,x0
    move    x0,y:lp_b2_\name
    move    x0,y:lp_a2_\name
    move    x0,y:lp_a1_\name
    move    x0,y:lp_b1_\name
    move    #>(1./16.),x0
    move    x0,y:lp_scale_\name

```

```

    ENDM

```

```

INIT_LP2_PARAM MACRO      name

```

```

    move    #0,x0
    move    x0,y:lp_a2_\name
    move    x0,y:lp_a1_\name
    move    #>0.5,x0
    move    x0,y:lp_b2_\name
    move    #>1.,x0
    move    x0,y:lp_b1_\name

```

0143

```

move    #>(0.25/1) x0
move    x0,y:lp_scale_\name

```

```

ENDM

```

```

;
;   assumes:
;       m0,m4,m5 = -1
;
;

```

```

LP  MACRO      name

```

```

    move    b,x0
    ori     #8,mr
    move    #lp_scale_\name,r4
    move    #lp_w_\name,r0

```

```

    mpy     x0,y0,b      x:(r0)+,x0
    mac     -x0,y0,b      x:(r0)-,x1
    macr    -x1,y0,b      x1,x:(r0)+
    mac     x0,y0,b      b,x:(r0)-
    macr    x1,y0,b
    andi    #$F7,mr
    asl     #4,b,b

```

```

ENDM

```

```

Y:(r4)+,y0      ; x0 = x(n), y0 = 0.5
Y:(r4)+,y0      ; x0 = w(n-2), y0 = a2
Y:(r4)+,y0      ; x1 = W(n-1), y0 = a1
Y:(r4)+,y0      ; w(n-2) = w(n-1), y0 = b2
Y:(r4)+,y0      ; w(n-1) = a, y0 = b1

```

```

HP_SCALE_FACTOR EQU 6
HP_SCALE_DIVIDE EQU (1<<HP_SCALE_FACTOR)

```

```

;
;
;   chamberlin hi-pass (from lopass)
;
;       q also defines gain
;       input is scaled to avoid internal clipping
;
;   f = 2*sin(w/2)  f only valid when less than 1.
;
;   0 < q < 1
;
;   implemented as a direct form biquad with coefficients and data scaled.
;   needs headroom for gain
;

```

```

DECLARE_HP MACRO      name,fc,qc,scale

```

```

    org xi:
    hp_s2_\name      ds 1
    hp_y_\name        ds 1

    org yi:
    hp_scale_\name    dc (-0.5*scale)
    hp_fc_\name        dc fc
    hp_qc_\name        dc qc

```

```

    ENDM

```

```

COPY_HP_PARAM MACRO      from,to

```

```

    move    y:hp_scale_\from,x0
    move    x0,y:hp_scale_\to
    move    y:hp_fc_\from,x0
    move    x0,y:hp_fc_\to
    move    y:hp_qc_\from,x0
    move    x0,y:hp_qc_\to

```

```

    ENDM

```

```

;
;   assumes:
;       m0,m4,m5 = -1
;
;

```

```

HP MACRO      name

```

```

    move    #hp_scale_\name,r4
    move    #hp_s2_\name,r0
    move    b,x0
    ; a = x(n) * scale      y:(r4)+,y0      ; y0 = scale
    mpy     -y0,x0,a      x:(r0)+,x1
    ; b = s2(n-1) * fc      y:(r4)+,y0      ; y0 = fc
    mpyr    x1,y0,b      x:(r0),x0
    ; b = s2(n-1) * fc + y(n-1)      y:(r4)+,y1      ; y1 = qc
    add     x0,b
    ; a = x(n) - s2(n-1) * fc - y(n-1)
    sub     b,a      b,x:(r0)
    ; a = s1(n) = x(n) - s2(n-1) * fc - y(n-1) - qc * s2(n-1)
    macr    -x1,y1,a      x1,b
    move    a,x0
    ; b = s2(n) = s2(n-1) + fc * s1(n)
    macr    x0,y0,b      x:(r0)-,x0

```

0145

```
IF @DEF('LO')
tfr    x0,b      b,x:(r0)
ELSE
tfr    a,b      b,x:(r0)
ENDIF
asl    b

ENDM
```

0146

```

AP_SCALE_FACTOR EQU 3
AP_SCALE_DIVIDE EQU (1<<AP_SCALE_FACTOR)

```

```

;
; gamma drives critical frequency
;
; beta defines Q
;

```

```

DECLARE_AP MACRO name,gamma,beta

```

```

    org xi:
ap_y\_name      dc 0
                dc 0
ap_x\_name      dc 0
                dc 0

    org yi:
ap_gamma\_name   dc gamma
ap_beta\_name    dc beta
ap_v\_name       dc 0
ap_w\_name       dc 0

```

```

ENDM

```

```

COPY_AP_PARAM MACRO from,to

```

```

    move y:ap_gamma\_from,x0
    move x0,y:ap_gamma\_to
    move y:ap_beta\_from,x0
    move x0,y:ap_beta\_to

```

```

ENDM

```

```

INIT_AP_PARAM MACRO name

```

```

ENDM

```

```

;
; input data is in b
; input is scaled by 0.5 to prevent internal clipping in the all-pass
;
; computes all-pass:
;
;  $v(n) = \gamma * y(n-1) + y(n-2) - \gamma * v(n-1)$ 
;  $w(n) = \gamma * x(n-1) + x(n-2) - \gamma * w(n-1)$ 
;  $y(n) = w(n) + \beta * x(n) - \beta * v(n)$ 
;
; assumes:
; m0,m4,m5 = -1
;
;

```

```

AP MACRO name

```

```

    asr b #ap_y\_name,r0
    move #ap_gamma\_name,r4
    move #ap_v\_name,r5
    move b,x0
    move x:(r0)+,a y:(r4)+,y0
    move x:(r0)+,x1 y:(r5)+,y1
; compute v(n)
    mac x1,y0,a x:(r0)+,b
    mac -y1,y0,a x:(r0),x1 y:(r5)-,y1
; compute w(n), v(n) is now in a
    mac x1,y0,b a,x1 a,y:(r5)+
    mac -y1,y0,b x:(r0),a y:(r4)+,y0
; compute y(n), w(n) is now in b
    mac -x1,y0,b x0,x:(r0)- b,y:(r5)-
    mac x0,y0,b a,x:(r0)-

```

0147

b  
b

```
x: (r0) -, a      b, y1
a, x: (r0) +
y1, x: (r0) -
```

b, y1

[illegible]

[illegible]